Correct Transformation of High-Level Models into Time-Triggered Implementations (DRAFT)

EPFL IC IINFCOM RiSD Technical Report
№ 218459
http://infoscience.epfl.ch/record/218459

Hela Guesmi (CEA LIST), Simon Bliudze (EPFL),
Belgacem Ben Hedia (CEA LIST),
Saddek Bensalem (Verimag), Mathieu Jan (CEA LIST),
Briag Lenabec (CEA LIST)

October 27, 2016

Abstract: A number of component-based frameworks have been proposed to tackle the complexity of the design of concurrent software and systems and, in particular, to allow modelling and simulation of critical embedded applications. Such design frameworks usually provide a capability for automatic generation of C++ or Java code, which has to be compiled for the selected target platform. Thus, guaranteeing hard real-time constraints is, at best, difficult. On the other hand, a variety of Real-Time Operating System (RTOS), in particular, those based on the Time-Triggered (TT) paradigm, guarantee the temporal and behavioural determinism of the executed software. However, such TT-based RTOS do not provide high-level design frameworks enabling the scalable design of complex safety-critical real-time systems.

In this paper, we combine advantages of the two approaches, by deriving correct-by-construction TT implementations from high-level componentised models. We present an automatic semantics-preserving transformation from RT-BIP (Real-Time Behaviour-Interaction-Priority) to PharOS—a safety-oriented RTOS, implementing the TT paradigm. The transformation has been implemented; we prove its correctness and illustrate it with a realistic case-study.

```
@TechReport{GBBH+16-BIP-to-TT,
  author      = {Guesmi, Hela
                 and Bliudze, Simon
                 and Ben Hedia, Belgacem
                 and Bensalem, Saddek
                 and Jan, Mathieu
                 and Lenabec, Briag},
  title       = {Correct Transformation of High-Level Models into
                 Time-Triggered Implementations},
  institution = {EPFL IC IINFCOM RiSD},
  month       = oct,
  year        = 2016,
  number      = {EPFL-REPORT-218459},
  note        = {Available at: http://infoscience.epfl.ch/record/218459}
}
```

Contents

To do

Correct Transformation of High-Level Models into Time-Triggered Implementations (DRAFT)

Hela Guesmi[3], Simon Bliudze[2], Belgacem Ben Hedia[1], Saddek Bensalem[3], Mathieu Jan[1] and Briag Lenabec[1]

[1]CEA, LIST, PC 172, 91191 Gif-sur-Yvette, France, Email: firstname.lastname@cea.fr
[2]EPFL IC IINFCOM RiSD, Station 14, 1015 Lausanne, Switzerland, Email: simon.bliudze@epfl.ch
[3]Verimag, 38610 Gieres, France, Email: firstname.lastname@imag.fr

Abstract

A number of component-based frameworks have been proposed to tackle the complexity of the design of concurrent software and systems and, in particular, to allow modelling and simulation of critical embedded applications. Such design frameworks usually provide a capability for automatic generation of C++ or Java code, which has to be compiled for the selected target platform. Thus, guaranteeing hard real-time constraints is, at best, difficult. On the other hand, a variety of Real-Time Operating System (RTOS), in particular, those based on the Time-Triggered (TT) paradigm, guarantee the temporal and behavioural determinism of the executed software. However, such TT-based RTOS do not provide high-level design frameworks enabling the scalable design of complex safety-critical real-time systems.

In this paper, we combine advantages of the two approaches, by deriving correct-by-construction TT implementations from high-level componentised models. We present an automatic semantics-preserving transformation from RT-BIP (Real-Time Behaviour-Interaction-Priority) to PharOS—a safety-oriented RTOS, implementing the TT paradigm. The transformation has been implemented; we prove its correctness and illustrate it with a realistic case-study.

Keywords—Component-based, correct-by-construction, time-triggered, transformation

1 Introduction

The Time-Triggered (TT) paradigm for the design of real-time systems was introduced by Kopetz [17]. TT systems are based on a periodic clock synchronization in order to enable a TT communication and computation. Each subsystem of a TT architecture is isolated by a so-called *temporal firewall*. It consists of a shared memory element for unidirectional exchange of information between sender and receiver task components. It is the responsibility of the *TT communication system* to transport, by relying on the common global time the information from the sender firewall to the receiver firewall. The strong isolation provided by the temporal firewall is key to ensuring the determinism of task execution and, thereby, allowing the implementation of efficient scheduling policies.

Developing embedded real-time systems based on the TT paradigm is a challenging task due to the increasing complexity of such systems and the necessity to manage, already in the programming model, the fine-grained temporal constraints and the low-level communication primitives imposed by the temporal firewall abstraction. Several Real-Time Operating Systems (RTOS) implement the TT execution model, such as PharOS [3] and PikeOS [16]. However, they do not provide high-level programming models that would allow the developers to think on a higher level of abstraction and to tackle the complexity of large safety-critical real-time systems. Model-based design frameworks, such as RT-BIP [1] and SCADE [7], allow the specification, design and simulation of real-time systems. In particular, RT-BIP—a component-based framework for the design of real-time systems—allows verification of behavioural properties, such as deadlock-freedom, and lends itself well to model transformations.

To the best of our knowledge, few connections exist between high-level component-based design frameworks, allowing reasoning about application models and verification of their functional behaviour and TT execution platforms, which guarantee temporal determinism of the system.

In this work, we establish a link between the model-based design framework RT-BIP and PharOS—a safety-oriented RTOS implementing the TT paradigm. The generation of PharOS applications from high-level RT-BIP models is done by a two-step transformation. The first step transforms a generic RT-BIP model into a restricted one, which lends itself well to an implementation based on TT communication primitives. It has been the subject of a previous work [11]. The second step, which is the contribution of this paper, transforms the resulting model into a PharOS application. We have outlined this transformation in a previous work-in-progress publication [12]. In this paper, we present more details about this step, identify the key difficulties in defining this transformation, propose exhaustive solutions to address these difficulties and prove that this transformation is semantics-preserving. This translation is fully implemented, but due to the lack of space, we do not present the code generation tool.

The rest of the paper is organized as follows. Section 2 describes the related work. In Section 3, we provide the necessary background on RT-BIP and PharOS. Section 4 presents a formal computation model of PharOS application tasks. Section 5 details the transformation form RT-BIP to PharOS. Section 6 and Section 7 deal with the correctness of the transformation, while Section 8 presents the case study. Section 9 summarises the paper and discusses future work directions.

2 Related Work

A design framework based on UML diagrams and targeting the TT architecture (TTA) [18] is presented in [21]. This approach relies on a decomposition of a system into clusters and nodes to instantiate the communication mechanisms. It assumes the underlying TT protocol to implement the FlexRay standard [22]. The framework does not support the earlier architectural design phase, nor the verification at model level. It requires a backward association mechanism to link faulty runs obtained at the SystemC level to the UML model.

Methods relying on model transformations in order to automatically refine AADL models are presented in [6, 9]. They are defined in order to reduce the gap between models used for timing analysis and for code generation. However, these approaches do not rely on well-defined formal semantics.

Our approach pursues the same goals as previous work coupling synchronous languages with TT platforms: a tool-chain for code generation from Lustre [14] to the TT architecture [10] and an automatic transformation from SCADE to PharOS [4]. Both these works are limited to relatively simple temporal behaviours. Indeed, their source models define periodic functional behaviour of the system, with the key real-time constraint being the duration of the period. In contrast, in our approach, RT-BIP source models define real-time constraints of arbitrary complexity.

Two model transformation approaches for generating distributed implementations from non-real-time BIP models and RT-BIP models, are presented respectively in [5] and [23]. In these approaches, the initial model is transformed into a 3-layer model relying exclusively on simple message-passing interactions, which are implementable using basic message-passing primitives. The third layer of this 3-layer model is reused by the transformation proposed in this paper for conflict resolution. Another method for generating a mixed hardware/software system model for many-core platforms from a high-level non-real-time application model and a mapping between software and hardware components is presented in [8]. The above approaches take advantage of the BIP framework to build correct-by-construction implementations based on a single semantic framework. Nevertheless, they do not target the platforms based on TT execution model, thereby falling short of exploiting the strong temporal guarantees provided by the latter.

3 Background

3.1 The RT-BIP Component Framework

RT-BIP is a component framework for constructing systems by superposing three layers of modelling: Behaviour, Interaction, and Priority. The Behaviour layer consists of a set of *components* defined by timed automata [2] extended with data and C functions. Transition labels of a component automaton are called *ports*. *Interactions* are sets of ports used for synchronization. Thus, the Interaction layer describes all possible synchronisations among components as a set of interactions. The third layer defines priorities among interactions, providing a mechanism for conflict resolution. In this paper, we do not consider priorities. Thus, we only consider RT-BIP models obtained by composing components with interactions.

Before formally defining RT-BIP components and their semantics, we introduce some notations. For a variable x, denote $D(x)$ its domain (i.e. the set of all values possibly taken by x). A valuation on a set of variables X is a function $v : X \rightarrow \bigcup_{x \in X} D(x)$, such that $v(x) \in D(x)$, for all $x \in X$. We denote by $\mathcal{V}(X)$ the set of all possible valuations on X and by $G_X = \mathbb{B}^{\mathcal{V}(X)}$ the set of *Boolean guards* on X.

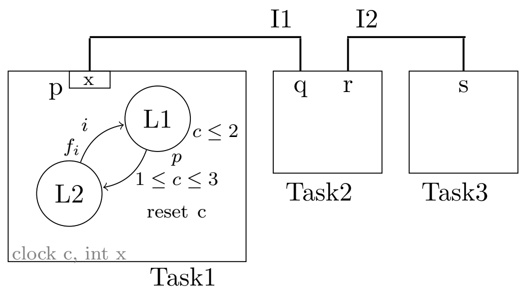


Figure 1: RT-BIP example

Definition 1 (Clock constraints)**.** *Let C be a set of clocks. The associated set G_C of clock constraints CC is defined by the following grammar:*

$$CC := \mathit{True} \mid \mathit{False} \mid c \sim a \mid CC \wedge CC\,,$$

with $c \in C$, $\sim \in \{\leq, =, \geq\}$ and $a \in \mathbb{Z}_+$. Notice that any guard CC can be written as:

$$CC := \bigwedge_{c \in C} l_c \leq c \leq u_c, \;\; \textit{where } \forall c \in C,\, l_c, u_c \in \mathbb{Z}_+ \cup \{+\infty\}\,. \tag{1}$$

Definition 2. *A* component *is a tuple $B = (L, P, X, C, T, tpc)$, where L is a finite set of locations; P is a finite set of ports; X is a finite set of local variables; C is a finite set of clocks; $T \subseteq L \times (P \times G_X \times G_C \times 2^C \times \mathcal{V}(X)^{\mathcal{V}(X)}) \times L$ is a finite set of transitions, each labelled with a port, two Boolean guards (on variables and on clocks), a set of clocks to be reset and a function updating a subset of variables of X; the function $tpc : L \to G_C$ assigns a* time progress condition *to each location, such that, for any $l \in L$, the constraint $tpc(l)$ is a conjunction of constraints of the form $c \leq u_c$.*

Figure 1, shows a model comprising three RT-BIP components `Task1`, `Task2` and `Task3`, composed by two binary interactions. The automaton of `Task1` is also shown in the figure. First consider `Task1` independently of the rest of the model and assume that the system reaches the state `L1` of `Task1` with $1 \leq c < 2$. Since $tpc(\texttt{L1}) = c \leq 2$, time can progress until $c = 2$ or, since the guard $1 \leq c \leq 3$ is also satisfied, transition p can be executed. If `L1` is reached with $c = 2$, the system cannot let the time progress and has to execute the transition p immediately. Finally, if $c > 2$, the time cannot progress and the system is blocked.

Definition 3 (Semantics of a component)**.** *The semantics of a component $B = (L, P, X, C, T, tpc)$ is defined as a Labelled Transition System (LTS) $(Q, P, \to)$, where $Q = L \times \mathcal{V}(X) \times \mathcal{V}(C)$ denotes the set of states of B and $\to\, \subseteq Q \times (P \cup \mathbb{R}_{\geqslant 0}) \times Q$ is the set of transitions defined as follows. Let (l, v_x, v_c) and (l', v'_x, v'_c) be two states, $p \in P$ and $\delta \in \mathbb{R}_{\geqslant 0}$.*

- ***Jump transitions**: We have $(l, v_x, v_c) \xrightarrow{p} (l', v'_x, v'_c)$ iff there exists a transition $\tau = (l, p, g_X, g_C, R, f, l') \in T$, such that $g_C(v_c) = g_X(v_x) = \mathit{True}$, $v'_x = f(v_x)$ and*

$$v'_c(c) = \begin{cases} 0\,, & \textit{for all } c \in R, \\ v_c(c) & \textit{for all } c \in C \setminus R. \end{cases}$$

- ***Delay transitions**: We have $(l, v_x, v_c) \xrightarrow{\delta} (l, v_x, v_c + \delta)$ iff $\forall \delta' \in [0, \delta]$, $tpc(l)(v_c + \delta') =$ True, where $(v_c + \delta)(c) \stackrel{def}{=} v_c(c) + \delta$, for all $c \in C$.*

A component B can execute a transition $\tau = (l, p, g_X, g_C, R, f_\tau, l')$ from a state (l, v_x, v_c) if its timing constraint is met by the valuation v_c. The execution of τ corresponds to moving from control location l to l', updating variables and resetting clocks of R. Alternatively, it can wait for a duration $\delta > 0$, if the time progress condition $tpc(l)$ stays *True*. This increases all the clock values by δ. Notice that execution of jump transitions is instantaneous; control location cannot change while time elapses.

Components communicate by means of interactions. An interaction is a synchronization between transitions of a fixed subset of components. It is possible only if all the participating components can execute the corresponding transitions. A formal definition can be found in [1], we omit it here for the sake of conciseness.

3.2 The PharOS platform

PharOS [3] is an operating system that allows designing, implementing and executing safety-critical multitasking applications based on the ΨC language.

The main building block of a PharOS [3] application is the TT task which is a set of successive jobs. Communication among tasks is through *temporal variables*, which are real-time data flows available to all tasks. Successive values of a temporal variable are produced by a single owner task at a predetermined temporal rhythm.

ΨC is a programming language designed for specifying different TT tasks and their synchronization points. It preserves the operational semantics of C, but adds time constraints to these semantics with the Ψ extension (this extension could be applied to any imperative programming language). C control flow graphs are automata, so C's instructions for control flow can be used to express sequencing of blocks, loops, and choices. The basic Ψ addition to C is the addition of the following instructions: `before`, `after`, and `advance` instructions that respectively add before and after constraints, and synchronization points.

- After instruction (`after`(d)): defines d as the relative release date of the following job;
- Before node (`before`(d)): defines d as the relative deadline of the preceding job;
- Advance node (`advance`(d))— also called synchronization point: combines `after`(d) and `advance`(d) instructions. It defines the absolute visibility date of the data produced by the job;

A PharOS application consists of a set of clock definitions followed by a set of task—also called agent—definitions. Recall that the set of parallel tasks communicates through (specific shared) variables. PharOS applications are characterised by the following abstract syntax:

$$\begin{aligned}
\text{Application} &::= \text{Clock}^{+}.\text{Agent}^{+}\,, \\
\text{Clock} &::= c = (\phi_c, P_c)\,, \\
\text{Agent} &::= \{\texttt{input}\ tv\}^{*}.\{\texttt{output}\ tv\}^{*}.\text{Body}\,, \\
\text{Body} &::= \{C\ \ code.[\texttt{after}(n)|\texttt{before}(n)|\texttt{advance}(n)]\ \texttt{with}\ \text{Clock}\}^{*}\,,
\end{aligned}$$

where c is a clock, with ϕ_c and P_c being respectively the *phase shift* and *period* of c (see the detailed definition below); tv is a temporal variable and $n \in \mathbb{Z}_+$ is a time step w.r.t. to an associated clock.

Clocks Clocks are variables used to describe the temporal behavior of the application. A clock defines a sequence of periodic instants called *activation instants*. These latter are used by the agents for describing timing constraints and synchronizations. Each clock c has an associated phase shift ϕ_c and a period P_c.

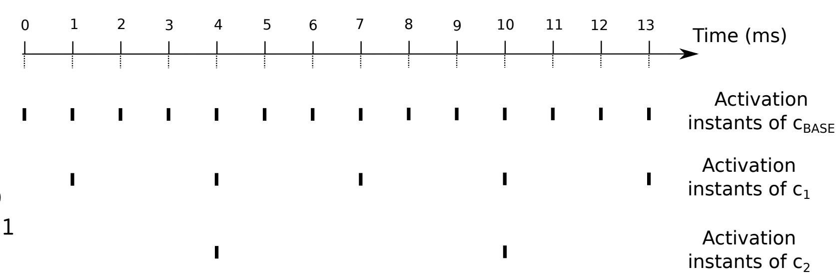
The global clock $c_{BASE} = (\phi_{BASE}, P_{BASE})$ is defined by its phase shift and period expressed in real time units, such as 1 second, 100 milliseconds etc. Formally, this clock defines a sequence of instants $(t_i)_{i\geq 0} = (i \cdot P_{BASE} + \phi_{BASE})_{i\geq 0}$. Other clocks $c = (\phi_c, P_c)$, are defined w.r.t. c_{BASE}, by putting $c = P_c * c_{BASE} + \phi_c$. Activation instants $(r_i)_{i\geq 0}$ of c are computed from those of c_{BASE} as follows:

$$r_i = (i \cdot P_c + \phi_c)P_{BASE} + \phi_{BASE} \,. \tag{2}$$

Although it is not used in this paper, the ΨC language also provides—for the designers' convenience—a possibility of defining new clocks in terms of clocks other than c_{BASE}. Figure 2b depicts activation instants of the clock c_{BASE} with period of one millisecond, a clock $c_1 = (1, 3)$ derived from c_{BASE} and a clock c_2 derived from c_1. Activation instants of c_1 are $1ms$, $4ms$, $7ms$ etc.. The ΨC code declaring the clocks of this example is shown in Figure 2a, where `gtc1` is the ΨC primitive declaring a global clock with milliseconds as the time unit.

```
clock cBASE = gtc1(0,1)
clock c1 = 3 * cBASE + 1
clock c2 = 2 * c1 + 1
```

(a)



(b)

Figure 2: Example of clocks and activation instants

In the remainder of this work, an instant t_i of c_{BASE} (resp. r_j of c) is referenced by its index i (resp. j). For example, in Figure 2, "instant 4 of c_{BASE}" refers to the physical activation instant $t_4 = 4ms$. Similarly, "instant 1 of c_1" refers to the instant $r_1 = 4ms$.

An instant r_i of clock $c = (\phi_c, P_c)$ can be mapped into an instant t_j of clock c_{BASE} by the function $conv^c_{c_{BASE}} : c \to c_{BASE}$, defined by letting

$$conv^c_{c_{BASE}}(r_i) = t_j \,, \quad \text{with} \quad j = i \cdot P_c + \phi_c \,. \tag{3}$$

Inversely, a global instant t_i of clock c_{BASE} can be mapped into an instant r_j of a derived clock c by using the function $conv_c^{c_{BASE}}$, defined by letting

$$conv_c^{c_{BASE}}(t_i) = r_j\,, \quad \text{with} \quad j = \left\lfloor \frac{i - \phi_c}{P_c} \right\rfloor . \tag{4}$$

For example, in Figure 2, the instant $r = 1$ of clock c_1 is mapped to the instant $conv_{c_{BASE}}^{c_1}(1) = 4$ of clock c_{BASE}. The instant $t = 5$ of clock c_{BASE} is mapped into instant $conv_{c_1}^{c_{BASE}}(5) = 1$ of clock c_1.

Agent An agent consists of an interface including declarations of input and output data flows (*temporal variables*) followed by a body. This latter describes the behavior of the agent through a block of timeless C code extended with *after*, *before* and *advance* statements. An `after`(d) (resp. `before`(d), `advance`(d)) with a clock $c = (\phi_c, P_c)$, defines the release (resp. deadline, synchronization) instant corresponding to d units of time after a reference instant. This reference instant corresponds to the absolute instant recording the visit of the last `after` or `advance` node.

```
body start
{
  // Job A
  ComputationA();

  // Job B
  after(1) with c;
  ComputationB();
  before(2) with c;

  // Job C
  ComputationC();
  advance(3) with c;

  // Job D
  ComputationD();
}
```

Figure 3: Example of body ΨC code

Code of Figure 3 describes the behavior of a task with four jobs (labelled A to D). In this example, all temporal constraints are defined over the same clock c. The release date of job B is one unit of time after the initial instant or previous advance constraint, i.e. `advance`(3), depending on the execution history. Two units of time later, job B must have ended. After the execution of the job C, communication take place since advance statement is reached. The visibility date of data produced by C is three units of time after the previous visit to the statement `after`(1).

A computation model of $PsiC$ language was provided in [19], where the behaviour of a task is specified using a directed graph, where arcs represent the successive jobs and nodes bear the temporal constraints. Nodes of the graph are of four types: After, before, advance and no-constraint nodes. This model is not at the same abstraction level as the $PsiC$ language since it does not hold clocks and thus does not provide the possibility of specifying constraints over different clocks. Also operational semantics are not provided.

In the next section, we provide a formal computation model of PsiC language and we express its semantics in terms of LTS.

4 Formal Computation Model of the $PsiC$ language

To define a formal translation from TT-BIP to PharOS application and to prove its correctness, we need to provide a formal definition of operational semantics of the target formalism. Moreover, we need that latter to be at the same abstraction level as the ΨC code, i.e. to specify constraints (release, deadline and synchronization shifts) over different clocks.

In this subsection, we present the Time-Constrained Automata (TCA) model as the computation model of PharOS applications. We detail first, how a PharOS task behavior can be presented (Definition 4) as a tuple in order to handle the multi-clock constraints. Then we provide its operational semantics(Definition 5).

A TCA automaton describes the behavior of a task, where nodes represent only the control locations and arcs are labelled by the triplets of constraints defining respectively the release, deadline and/or synchronization instants. Each component of such triplet is either $(-1, \perp)$, or a pair of a shift constraint and a clock over which this shift is defined. We denote by $M \stackrel{def}{=} (\mathbb{Z}_+ \times C) \cup \{(-1, \perp)\}$ the set of all such labels. When a label is $(-1, \perp)$, the corresponding constraint is not defined.

Given a shift $x \in \mathbb{Z}_+$ over a clock c, and a reference instant λ over the global clock c_{BASE}, we denote by $shift^c_{c_{BASE}} : c_{BASE} \times \mathbb{Z}_+ \to c_{BASE}$ the function computing the global instant corresponding to the desired shift as follows:

$$shift^c_{c_{BASE}}(\lambda, x) = conv^c_{c_{BASE}}(conv^{c_{BASE}}_c(\lambda) + x), \tag{5}$$

where $conv^c_{c_{BASE}}$ and $conv^{c_{BASE}}_c$ are defined in (3) and (4), respectively.

Definition 4 (TCA). *A TCA is a tuple (N, K, X, C, T) where N is a finite set of* nodes*, K is a finite set of* jobs*, X is a set of* local variables*, C is a set of clocks, comprising a real-time global clock c_{BASE} and other clocks derived from c_{BASE}, and $T = N \times G_X \times M^3 \times K \times \mathcal{V}(X)^{\mathcal{V}(X)} \times N$ is a set of transitions. Thus, a transition is a tuple $\tau = (n, g_X, m, k, f, n') \in T$ where:*

- *$g_X \in G_X$ is a Boolean guard on X;*
- *$m = ((r, c_r), (d, c_d), (s, c_s)) \in M^3$, is a triplet defining respectively, the release shift over clock c_r, the deadline shift over clock c_d and/or the synchronization shift over clock c_s. If $(s, c_s) \neq (-1, \perp)$, then $(d, c_d) = (s, c_s)$. If $(r, c_r) \neq (-1, \perp)$ and $(d, c_d) \neq (-1, \perp)$, then $shift^{c_r}_{c_{BASE}}(\lambda, r) \leq shift^{c_d}_{c_{BASE}}(\lambda, d)$ for any $\lambda \in c_{BASE}$ (i.e. if defined, the global release instant is always inferior than or equal to the global deadline instant);*

- $k \in K$ *is a job;*
- $f \in \mathcal{V}(X)^{\mathcal{V}(X)}$ *is an update function on variables in* X.

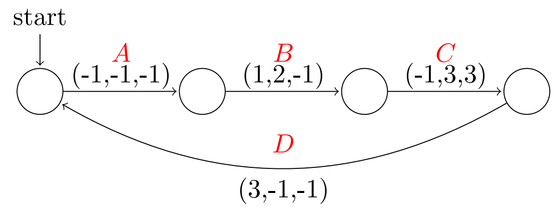


Figure 4: Alternative representation of the task behavior of Figure 3

The automaton presenting the task behavior of Figure 3 is shown in Figure 4. Since in the model of Figure 3 all constraints are defined over the same clock, we only show the first component, i.e. the shift, of each pair of the triplet-label. This triplet-label depends on timing instructions encompassing the job in the original body code. The label of the job A is $((-1, \perp), (-1, \perp), (-1, \perp))$ since in the original code, it is not preceded by an `after` instruction, nor succeeded by a `before` or `advance` instruction. Notice that, in the labels of job C, the deadline shift coincides with the corresponding synchronization shift, reflecting the fact that in the original behavior code, this job is succeeded by an `advance` instruction.

Defining the operational semantics of TCA automaton requires a notion of state. The state of a TCA automaton is described in four parts: the control state (i.e. control location), the state of the data variables, the state of the clock variables and the state of a delay variable (needed for absolute constraints computation). TCA semantics can be defined as a labelled transition system:

Definition 5 (Semantics of TCA)**.** *The semantics of a time-constrained automaton* (N, K, X, C, T) *is defined as a labelled transition system* $(Q, K, \rightarrow)$, *where* $Q = N \times \mathcal{V}(X) \times \mathcal{V}(C) \times \mathbb{R}_{\geqslant 0}$ *and* $\rightarrow \subseteq Q \times K \times Q$ *is the set of transitions, defined as follows. Let* $(n, v_x, v_c, v_{\lambda_{ref}})$ *and* $(n', v'_x, v'_c, v'_{\lambda_{ref}})$ *be two states, such that* $v_c \leq v'_c$ *for all* $c \in C$. *We have* $(n, v_x, v_c, v_{\lambda_{ref}}) \xrightarrow{k} (n', v'_x, v'_c, v'_{\lambda_{ref}})$ *iff there exists a transition* $(n, g_X, ((r, c_r), (d, c_d), (s, c_s)), k, f, n') \in T$ *such that :*

- $g_X(v_x) = True$,
- $v'_x = f(v_x)$,
- $v_c = v'_c$, *if* $((r, c_r), (d, c_d), (s, c_s)) = ((-1, \perp), (-1, \perp), (-1, \perp))$, *otherwise* $v_c \leq v'_c$,
- *if* $(r, c_r) \neq (-1, \perp)$, *then* $\forall c \in C \setminus \{c_{BASE}\}$, $shift^{c_r}_{c_{BASE}}(v_{\lambda_{ref}}, r) \leq conv^{c}_{c_{BASE}}(v'_c(c))$,
- *if* $(d, c_d) \neq (-1, \perp)$, *then* $\forall c \in C \setminus \{c_{BASE}\}$, $conv^{c}_{c_{BASE}}(v'_c(c)) \leq shift^{c_d}_{c_{BASE}}(v_{\lambda_{ref}}, d)$,

- $v'_{\lambda_{ref}} = \begin{cases} shift^{c_s}_{c_{BASE}}(v_{\lambda_{ref}}, s)\,, & \text{if } s \neq -1, \\ shift^{c_r}_{c_{BASE}}(v_{\lambda_{ref}}, r)\,, & \text{if } s = -1 \text{ and } r \neq -1, \\ v_{\lambda_{ref}}\,, & \text{if } s = -1 \text{ and } r = -1. \end{cases}$

An execution sequence of a TCA automaton is a sequence of transitions from different states of the system. It is defined as follows:

Definition 6 (Execution Sequence)**.** *A finite (resp. infinite) execution sequence of a time-constrained automaton (N, K, X, C, T) from an initial state $(n^0, v^0_x, v^0_c, v^0_{\lambda_{ref}})$ is a sequence of transitions:*

$$(n^i, v^i_x, v^i_c, v^i_{\lambda_{ref}}) \xrightarrow{k_i} (n^{i+1}, v^{i+1}_x, v^{i+1}_c, v^{i+1}_{\lambda_{ref}})\,,$$

where $k_i \in K$ and $i \in [1, n]$ such that $n \in \mathbb{Z}_+$.

Lemma 1. *Given a job $k_i \in K$ with $i \in [1, n]$, such that $q_i \xrightarrow{k_i} q_{i+1}$, then $\forall c \in C$,*

$$conv^c_{c_{BASE}}(v^i_c(c)) \leq conv^c_{c_{BASE}}(v^{i+1}_c(c))\,,$$

where v^i_c denotes the clock valuation component of the corresponding state q_i.

Notice that the combination of Lemma 1 with Definition 5 allows to extend implicitly constraints of a transition to its succeeding or preceding ones, i.e. given a transition $\tau_i = (n, g_X, ((r, c_r), (d, c_d), (s, c_s)), k_i, f, n') \in T$, where $(r, c_r) \neq (-1, \perp)$ and $(d, c_d) \neq (-1, \perp)$, the deadline shift (d, c_d) applies implicitly to all preceding transitions while the release shift (r, c_r) applies to all succeeding transitions. That is, for jobs $k_{i-1}, k_i, k_{i+1} \in K$ with $i \in [1, n]$, such that: $q_{i-1} \xrightarrow{k_{i-1}} q_i \xrightarrow{k_i} q_{i+1} \xrightarrow{k_{i+1}} q_{i+2}$, we have $\forall c \in C$,

$$conv^c_{c_{BASE}}(v^{i-1}_c(c)) \leq conv^c_{c_{BASE}}(v^i_c(c)) \leq conv^c_{c_{BASE}}(v^{i+1}_c(c)) \leq shift^{c_d}_{c_{BASE}}(v^i_{\lambda_{ref}}, d)$$

and

$$shift^{c_r}_{c_{BASE}}(v^i_{\lambda_{ref}}, r) \leq conv^c_{c_{BASE}}(v^{i+1}_c(c)) \leq conv^c_{c_{BASE}}(v^{i+2}_c(c))\,,$$

where v^i_c (resp. $v^i_{\lambda_{ref}}$) denotes the clock valuation component (resp. the component of λ_{ref} valuation) of the corresponding state q_i.

5 From RT-BIP to PharOS

We transform an RT-BIP model into a PharOS application in two steps (cf. Figure 5).

$$\boxed{\text{RT-BIP}} \xrightarrow[\text{[11]}]{\text{step 1}} \boxed{\text{TT-BIP}} \xrightarrow{\text{step 2}} \boxed{\text{TCA}}$$

Figure 5: From RT-BIP to the TCA model: a 2-step transformation.

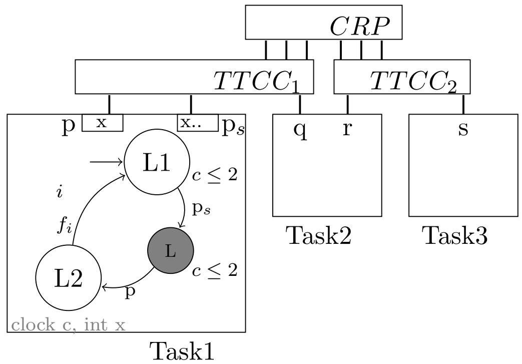


Figure 6: TT-BIP model derived from the example of Figure 1

Step 1: RT-BIP to TT-BIP. This transformation—published in [11] and proven to be semantics-preserving—consists in adapting the initial model to comply with the TT communication pattern. The obtained model—called TT-BIP—is a structural restriction of RT-BIP respecting the TT paradigm.
In this transformation, components are grouped into tasks by following a user-defined task mapping. Each interaction involving several tasks is replaced by a protocol, implemented by a dedicated *TT communication component* (TTCC). Interactions between task components and TTCC components are binary synchronisations with unidirectional data transfer. A *conflict resolution protocol* (CRP) component is used for resolving conflicts between interactions [5]. A task component can define several clocks, but we consider that each transition within a task can be labelled only by a single-clock constraint and all clocks are never reset. CRP and TTCC components rely on a common clock, which is computed as being the greatest common divisor of all tasks clocks and is never reset. The set of all ports is partitioned into *send* (P_s), *receive* (P_r) and *internal ports* (P_i). In the TT-BIP model, if a send port is enabled at some global state, then all its receive ports are also enabled at that state. Figure 6 shows the obtained TT-BIP model after applying step 1-transformation to the example of Figure 1.

Step 2: TT-BIP to TCA. In this step, we transform the TT-BIP model—obtained in Step 1—into a TCA automaton.

In the remainder of this section, we discuss the challenges of the transformation in Step 2, define the semantics of TCA in terms of LTS and the formal transformation rules allowing to derive TCA from TT-BIP models.

5.1 Transformation challenges

Moving from absolute to relative constraints. In TT-BIP, all constraints are defined in terms of absolute clock values. On the contrary, TCA nodes bear only relative constraints, i.e. as an increment to previous $\triangleright_d$ or $\diamond_d$ node.

In order to address this issue, we make use of the variable λ^c_{ref}. It is initiated to zero and updated whenever an `after` or an `advance` node is instantiated. Thus, it stores the valuation of the clock c in the last visited `after` or `advance` node. Relative constraint $d_{relative}$ is computed from its corresponding absolute constraint $d_{absolute}$ following this formula:

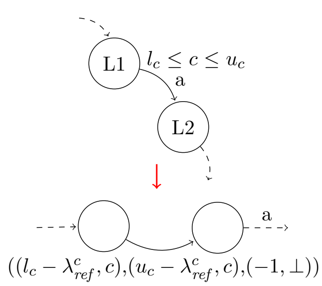
$$d_{relative} = d_{absolute} - \lambda^c_{ref} \, . \tag{6}$$

Mapping of timing constraints. Both RT-BIP and TT-BIP models are based on an abstract notion of time. In particular, actions that correspond to the computational steps (jump transition) of the system are considered to be atomic and have zero execution times. Thus only start instants of these actions have associated timing constraints. Delay steps, are specified by timing progress conditions (*tpc*) indicating whether time can progress at a given state of the system. However in TCA models, actions are considered to have both a release date and a deadline. These dates can be easily specified by using `after` and `before` instructions of the ΨC language, which correspond to a release and deadline labels in the TCA model presented in Section 4.

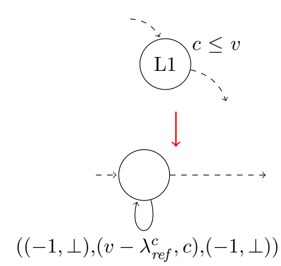
This issue can be addressed by transforming each step of a TT-BIP automaton (computational or delay step) into one or more actions in the final TCA automaton as follows.

A computational step τ having a timing constraint of the form $l_c \leq c \leq u_c$ in TT-BIP, has only its start instant constrained. It is supposed to start at any instant between l_c and u_c. In order to keep the same semantics in TCA, we can insert a job to mark the beginning of each transition. The label of this job is the following: $((l_c - \lambda^c_{ref}, c), (u_c - \lambda^c_{ref}, c), (-1, \perp))$. It has no update actions and defines a release and deadline shifts. This ensures that the following job will start at an instant respecting the constraint of τ. After the initiated job, the actions of the original transition τ can be executed in a new job depending on whether the original transition corresponds to internal computation or communication. The example in Figure 7a illustrates the mapping rule of a transition having a constraint of the form $l_c \leq c \leq u_c$.

Delay steps are constrained by the time-progress conditions of the form $c \leq v$. In TCA, these can be encoded by a loop job labelled by $((-1, \perp), (v - \lambda^c_{ref}, c), (-1, \perp))$, since in the original model the start instant of the delay step is not specified and only its deadline d is defined (see Figure 7b).



(a) Computational step constraint



(b) Delay step constraint

Figure 7: Mapping of constraints

Communication mapping. In TT-BIP, all tasks are related to communication components via binary interactions, which provide unidirectional data transfer and synchronization between sending and receiving actions of, respectively, the sender and the receiver components. In TCA, the communication is performed through the temporal variable model. New values of temporal variables are made visible at each of the synchronization points of the sender. These new values are consulted when the current time of receivers is greater than or equal to the visibility date of the new values. In our transformation two requirements need to be satisfied: (1) the receiver must consult an updated temporal variable (i.e. after the sending action of the sender task) and (2) we need to respect communication semantics of the initial model, i.e. the synchronisation between write and read actions.

We generate TCA synchronization points (`advance` instructions in ΨC language) that depend on whether the TT-BIP transition is triggered by a send, receive or an internal port. For each communicating transition in the original model, we instantiate —after jobs guaranteeing respect of timing constraint (cf. Figure 7) —a job containing in its triplet-label the synchronization component $(1, c_{fg})$, where c_{fg} is a fine-grained clock.

For example, consider —in the original model —a sender and a receiver components having the same clock. Suppose they are meant to communicate in the same instant t in TT-BIP model. We can define a finer clock c_{fg}, allowing the instantiation of synchronisation points (send and receive at $t + \epsilon$). For example, consider the time line in Figure 8. The visibility instant of the sender data is $4*t+1$ of the clock c_{fg}. The receiver will consult these data in the instant $4*t+2$ of the clock c_{fg}. In this example both requirements cited above are satisfied:

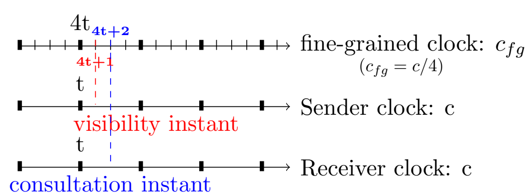


Figure 8: Example of `advance` nodes defined over c_{fg}

(1) the sender updates the variable before the receiver consults it; (2) when considering the initial clock c over which the synchronization instant t was defined, these write and read instants can be approximated to t since the instant $t+1$ over c is still not reached.

In order to address this challenge for an arbitrary initial model without resorting to ad hoc solutions, we propose the following solution. We define a fine-grained clock $c_{fg} = c_{TTCC}/2$ where c_{TTCC} is the clock of TTCC components in TT-BIP model. All synchronization points (i.e. the third component of the triplet-label) are defined over this clock. To each sending action we associate a job labelled by $((-1, \perp), (1, c_{fg}), (1, c_{fg}))$ (i.e. `advance`(1) statement in the $PsiC$ code). Note that this job is instantiated after guaranteeing the respect of timing constraint in the original model (i.e. after instantiating jobs as in Figure 7). We add a Boolean flag in each transferred message, which will allow testing the freshness of the message. The sender automaton changes the state of this flag whenever a sending transition is executed. The receiver automaton, has a local flag used as reference. The value of that flag is set to the value of the flag of the last received message. To each receiving transition, we associate a loop job labelled by $((-1, \perp), (1, c_{fg}), (1, c_{fg}))$ corresponding to successive reception attempts until the message is detected to be fresh. That is until the value of the local flag is different from the value of the flag of the message. Since in the TT-BIP model, all the receive-ports of an interaction are enabled if the send port is enabled, we can be sure that the receiving job in the obtained TCA automaton will occur at latest one instant after the sending one over the clock c_{fg}. The synchronization requirement over the clock c_{TTCC} is thus satisfied.

5.2 Formal translation

In this section, we present a formal definition of the transformation outlined in Section 5.1

Definition 7. *Let $B = (L, P, X, C, T, tpc)$ be a TT-BIP component with $P = P_i \cup P_s \cup P_r$ (see the opening of Section 5). For $l \in L$, we denote $P_l = \{p \in P \mid l \xrightarrow{p}\}$ the set of ports enabled in l. The TCA corresponding to B is defined by putting $TCA_B = (N, K, X \cup X' \cup Y, C \cup \{c_{BASE}, c_{fg}\}, T')$, with*

$$N = \{N_l^0 | l \in L\} \cup \{l_p | l \in L, p \in P_l\} \cup \{N_{l_p}^1 | l \in L, p \in P_l \cap P_r\}$$

$K = P \times \{send, receive, internal\}$, $X' = \{flag^p \mid p \in P_s \cup P_r\}$ and Y are the sets of flags and variables used for managing communication and the set of transitions T' defined as follows. For each transition $\tau = (l, p, g_X, tc, r, f, l') \in T$, with $tc = (lb \le c \le ub)$ and $tpc(l) = (c' \le v)$, $c, c' \in C$, the set T' comprises the following transitions:

$$\tau_l^{tpc} = (N_l^0, True, ((-1, \perp), (v - \lambda_{ref}^{c'}, c'), (-1, \perp)), (p, internal), id, N_l^0),$$

$$\tau_{l_p}^0 = \begin{cases} (N_l^0, g_X, ((lb - \lambda_{ref}^c, c), (ub - \lambda_{ref}^c, c), (-1, \perp)), (p, internal), f_{flag}^p, l_p), & \text{if } p \in P_s, \\ (N_l^0, g_X, ((lb - \lambda_{ref}^c, c), (ub - \lambda_{ref}^c, c), (-1, \perp)), (p, internal), id, l_p), & \text{if } p \in P_i \cup P_r, \end{cases}$$

$$\tau_{l_p}^1 = \begin{cases} (l_p, True, ((-1, \perp), (1, c_{fg}), (1, c_{fg})), (p, internal), id, N_{l_p}^1), & \text{if } p \in P_s, \\ (l_p, \neg g_{fresh}^p, ((-1, \perp), (1, c_{fg}), (1, c_{fg})), (p, internal), id, l_p), & \text{if } p \in P_r, \end{cases}$$

$$\tau_{l_p} = \begin{cases} (l_p, True, ((-1, \perp), (-1, \perp), (-1, \perp), (p, internal), f, N_{l'}^0), & \text{if } p \in P_i, \\ (N_{l_p}^1, True, ((-1, \perp), (-1, \perp), (-1, \perp)), (p, send), f, N_{l'}^0), & \text{if } p \in P_s, \\ (l_p, g_{fresh}^p, ((-1, \perp), (-1, \perp), (-1, \perp)), (p, receive), f \circ f_{update}^p, N_{l'}^0), & \text{if } p \in P_r. \end{cases}$$

where $\lambda_{ref}^{c'} = conv_c^{c_{BASE}}(\lambda_{ref}, c')$, $\lambda_{ref}^c = conv_c^{c_{BASE}}(\lambda_{ref}, c)$, id is the identity function, $f_{flag}^p : \mathcal{V}(X') \to \mathcal{V}(X')$ is the function that flips the value of the Boolean variable $flag^p$ before sending a message, g_{fresh}^p is the guard verifying whether the value of $flag^p$ is different from that contained in the received message, $f_{update}^p : \mathcal{V}(X \cup Y) \to \mathcal{V}(X)$ is the function updating local variables according to received values if $p \in P_r$ and c_{fg} is the clock having half of the period of the smallest clock (i.e. clock c_{TTCC}) in the initial RT-BIP model (cf. the third paragraph of Section 5.1).

Notice that the domain and co-domain of the function f in the transition τ above are given by $f : X \to X$. Hence the composition $f \circ f_{update}^p$ is well-defined.

For all types of triggering ports in the original transition, the instantiated τ_l^{tpc} transition, maps the delay transition modelled by the *tpc* constraint associated to the source state of the original transition. It does not execute any action, but allows waiting as long as the deadline v is not reached. Transition $\tau^0_{l_p}$ constrains the release date of the original transition actions and is guarded by g_X. For internal and receive ports, it does not execute any actions; for a send port p, it executes f^p_{flag}, flipping the message flag. Transition $\tau^1_{l_p}$ is instantiated only for send and receive ports. It allows the synchronization (communication) if the labelling port is a send port. If the labelling port is a receive port, $\tau^1_{l_p}$ is a loop transition on the node l_p consisting in synchronization attempts while the message is not fresh. Transition τ_{l_p}—for internal or send triggering ports—executes actions of the original transition and it has no guard nor timing constraint. If the labelling port of the original transition is a receive port, the transition τ_{l_p} marks the end of synchronization attempts when the received message is detected to be fresh (through the guard g^p_{fresh}). It also updates (through f_{update}) local variables according to received message before execution the function f of the original transition (i.e. it executes the composition $f \circ f_{update}$). In case when the labelling port is a send (resp. receive) port, the transition τ_{l_p} is labelled by a different label from the rest of transitions which is (p, send) (resp. $(p, \mathit{receive})$).

Figure 9 displays the sets of transitions in the obtained TCA automaton corresponding to a transition shown in Figure 9a and depending on the type of port p.

In the following paragraph, we provide more details about encoding of f^p_{flag} and g^p_{fresh}. And we show how $\tau^1_{l_p}$ and τ_{l_p} allow the reception of the actual message.

Encoding of communication details. Consider a receive port p and the corresponding local Boolean variable flag^p. Denote the Boolean flag of the message received through p by flag^{msg}. The guard g^p_{fresh} is defined by putting

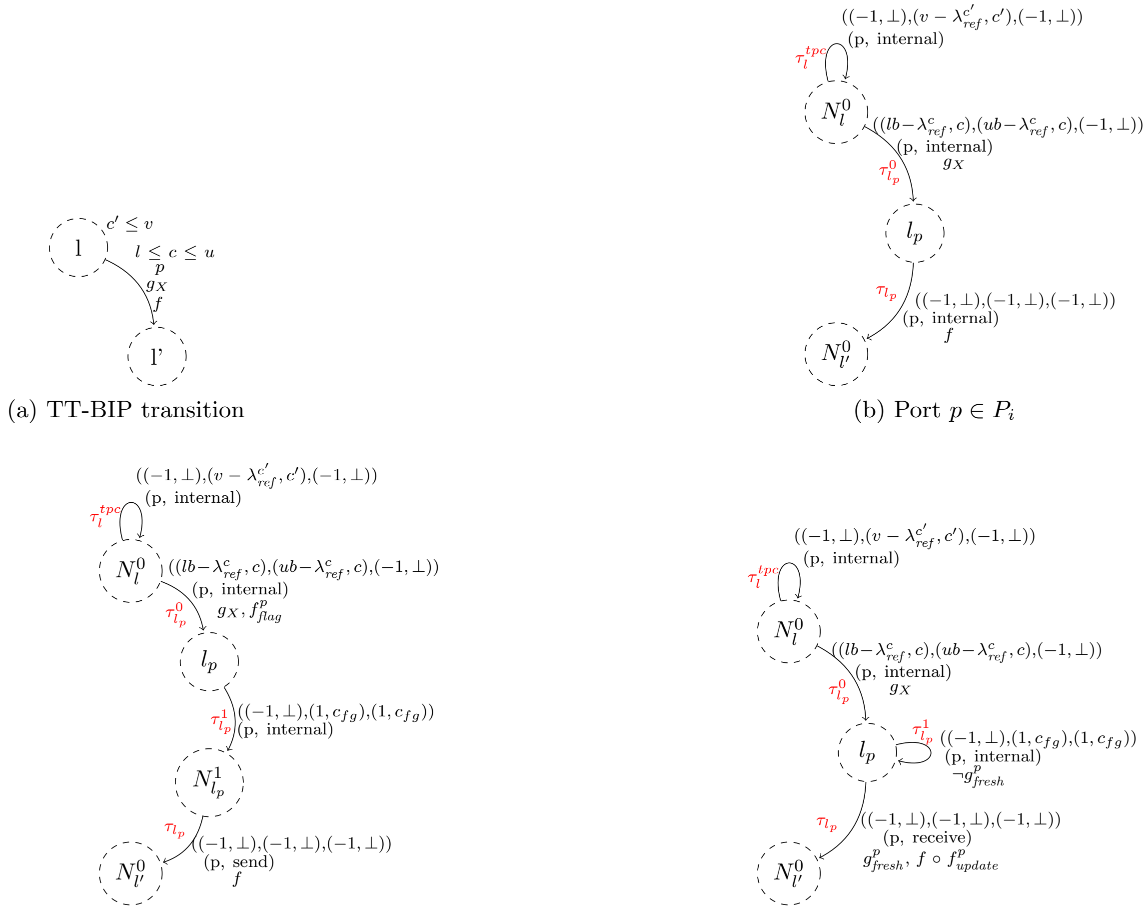
$$g^p_{\mathit{fresh}} \stackrel{def}{=} (\mathit{flag}^p \neq \mathit{flag}^{msg}).$$

By construction, flag^p and flag^{msg} are initiated to zero. Thus, initially, we have $\neg g^p_{\mathit{fresh}} = \mathit{True}$ and the loop transition $\tau^1_{l_p}$ is enabled. This transition will perform a communication attempt (through an `advance`(1) node) with no actions on local variables. Each communication attempt leads to the implicit update of the guard g_{fresh} depending on the flag of the received message. If the sender has sent a new message—through its corresponding transition $\tau^1_{l_{p'}}$—it should have performed the function $f^{p'}_{\mathit{flag}}$ in order to change the value of flag^{msg} with:

$$f^{p'}_{\mathit{flag}} = (\mathit{flag}^{p'} := \neg \mathit{flag}^{p'}) \,.$$

Recall that $\mathit{flag}^{p'}$ is a local variable of the sending component, whereof the value is incorporated into the message. This is the value that, upon reception of the message by the receiving component, we denote flag^{msg}. Thus, upon reception of the message g^p_{fresh} evaluates to *True*, enabling the transition τ_{l_p} in the receiver automaton. Otherwise, if the sender did not send the new message yet, g^p_{fresh} evaluates to *False* and the transition $\tau^1_{l_p}$ is again enabled.

Notice also that among the values contained in the message, only flag^{msg} is tested after execution of transition $\tau^1_{l_p}$. This value is only used to evaluate the freshness of each received message.



(a) TT-BIP transition

(b) Port $p \in P_i$

(c) Port $p \in P_s$

(d) Port $p \in P_r$

Figure 9: A TT-BIP transition (a) and the corresponding set of TCA transitions (b)–(d)

Since the transition τ_{l_p} of the receiver automaton is executed when the received message is fresh, it is in charge of making local copies of message variables through the function f^p_{update} before executing the function f of the initial transition. The function f^p_{update} copies also the value of flag^{msg} into flag^p, thereby also changing the value of g^p_{fresh} from *True* to *False*.

6 Correctness of the transformation

In order to prove the correctness of the transformation from TT-BIP to TCA, we have to show that the corresponding semantic LTS are equivalent. This is illustrated in Figure 10, where F denotes the transformation from TT-BIP to TCA (Definition 7), G_1 and G_2 denote the corresponding LTS semantics.

We define observational equivalence between transition systems based on the classical notion of weak bisimilarity [20], where some transitions are considered unobservable.

We will use the following notation. Consider a binary relation $R \subseteq X \times Y$. For $x \in X$, we denote $R(x) \stackrel{def}{=} \{y \in Y \mid (x, y) \in R\}$.

Definition 8. *(LTS relations) Let $A = (Q_A, P_A, \underset{A}{\longrightarrow})$ and $B = (Q_B, P_B, \underset{B}{\longrightarrow})$ be two LTS. Given a relation $\beta \subseteq P_A \times P_B$, we write $q \overset{\beta}{\underset{A}{\longrightarrow}} q'$, for $q \in Q_A$, iff there exists $a \in P_A$, such that $q \overset{a}{\underset{A}{\longrightarrow}} q'$ and a is* not *related by β to any label in P_B, i.e. $\beta(a) = \emptyset$. The notation $q \overset{\beta}{\underset{B}{\longrightarrow}} q'$, for $q \in Q_B$, is defined symmetrically.*

A weak simulation *over A and B, is a pair of relations $R \subseteq Q_A \times Q_B$ and $\beta \subseteq P_A \times P_B$, such that:*

$$\forall (q, r) \in R, \forall a \in P_A, \left(\beta(a) \neq \emptyset \wedge q \overset{a}{\underset{A}{\longrightarrow}} q' \Longrightarrow \exists (a, b) \in \beta : \exists (q', r') \in R : r \xrightarrow[B]{\beta^* b \beta^*} r'\right)$$

and

$$\forall (q, r) \in R, \left(q \overset{\beta}{\underset{A}{\longrightarrow}} q' \Longrightarrow \exists (q', r') \in R : r \overset{\beta^*}{\underset{B}{\longrightarrow}} r'\right).$$

A weak bisimulation *over A and B is a pair of relations $R \subseteq Q_A \times Q_B$ and $\beta \subseteq P_A \times P_B$, such that both (R, β) and (R^{-1}, β^{-1}) are weak simulations. We say that A and B are* weakly bisimilar w.r.t. *$\beta \subseteq P_A \times P_B$, denoted $A \sim_\beta B$, if there exists $R \subseteq Q_A \times Q_B$ total on both Q_A and Q_B, such that (R, β) is a weak bisimulation.*

$$\begin{tikzcd}
\boxed{\text{TT-BIP}} \arrow[r, "F"] \arrow[d, "G_1"] & \boxed{\text{TCA}} \arrow[d, "G_2"'] \\
\boxed{\text{LTS}} \arrow[r, phantom, "\sim_\beta"] & \boxed{\text{LTS}}
\end{tikzcd}$$

Figure 10: Translation functions

Let $B = (L, P, X, C, T, tpc)$ be a TT-BIP component. We need to prove equivalence between $G_1(B)$ and $G_2(F(B))$. To this end, we define the following relation on labels of the two LTS:

$$\beta = \{(p, (p, send)) \mid p \in P_s\} \cup \{(p, (p, receive)) \mid p \in P_r\}. \tag{7}$$

NB: In this draft version of the report, this theorem is not yet finilised.

Theorem 1. *The LTSs $G_1(B)$ and $G_2(F(B))$ are weakly bisimilar w.r.t. β, i.e. $G_1(B) \sim_\beta G_2(F(B))$.*

Proof. Let $G_1(B) = (Q_B, P, \underset{B}{\longrightarrow})$ and $G_2(F(B)) = (Q_{TCA}, K, \underset{TCA}{\longrightarrow})$. Recall (Definition 3) that state space Q_B has three components: control location, clock and variable valuations while the state space Q_{TCA} (Definition 5) has an extra fourth component—besides the three components previously cited—consisting in the valuation of the delay λ_{ref}. For a given state q, we will denote $v_c(q)$ (resp. $v_x(q)$) its clock (resp. variable) valuation component. For a given state $q \in Q_{TCA}$, we will denote $v_{\lambda_{ref}}(q)$ the corresponding valuation of λ_{ref}.

Below, we will use variables q_B, r_B, ranging over Q_B, and q_{TCA}, r_{TCA}, ranging over Q_{TCA} and denote their respective components as follows:

$$q_B = (l, v_x(q_B), v_c(q_B))\,, \qquad r_B = (l', v_x(r_B), v_c(r_B))\,,$$
$$q_{TCA} = (n, v_x(q_{TCA}), v_c(q_{TCA}), v_{\lambda_{ref}}(q_{TCA}))\,, \quad r_{TCA} = (n', v_x(r_{TCA}), v_c(r_{TCA}), v_{\lambda_{ref}}(r_{TCA}))\,.$$

We define the relation $R \subseteq Q_B \times Q_{TCA}$ as follows:

$$R = \left\{ (q_B, q_{TCA}) \;\middle|\; \begin{array}{c} n \in \{N_l^0\} \cup \{l_p\}_{p \in P_l} \cup \{N_{l_p}^1\}_{p \in P_l \cap P_s}\,, \\ v_c(q_B) = v_c^*(q_{TCA}), \\ v_x(q_B) = v_x^*(q_{TCA}) \end{array} \right\} \tag{8}$$

where v_c^* (resp. v_x^*) is the restriction of v_c (resp. v_x) to the set of clocks C (resp. variables X). That is the valuation function v_c^* (resp. v_x^*) is defined only over clocks (resp. variables) which are common between B and $F(B)$, i.e. excluding clocks c_{BASE} and c_{fg} (resp. variables $X' \cup Y$) of $F(B)$.

The following four assertions prove that (R, β) is a weak bisimulation:

(i) $\forall (q_B, q_{TCA}) \in R\,,$

$$q_B \xrightarrow[B]{\beta} r_B \implies \exists (r_B, r_{TCA}) \in R : q_{TCA} \xrightarrow[TCA]{\beta^*} r_{TCA}\,,$$

(ii) $\forall (q_B, q_{TCA}) \in R\,,$

$$q_{TCA} \xrightarrow[TCA]{\beta} r_{TCA} \implies \exists (r_B, r_{TCA}) \in R : q_B \xrightarrow[B]{\beta^*} r_B\,,$$

(iii) $\forall (q_B, q_{TCA}) \in R\,, \forall p \in P\,,$

$$\beta(p) \neq \emptyset \wedge q_B \xrightarrow[B]{p} r_B \implies \exists (p,k) \in \beta : \exists (r_B, r_{TCA}) \in R : q_{TCA} \xrightarrow[TCA]{\beta^* k \beta^*} r_{TCA}\,,$$

(iv) $\forall (q_B, q_{TCA}) \in R\,, \forall k \in K\,,$

$$\beta^{-1}(k) \neq \emptyset \wedge q_{TCA} \xrightarrow[TCA]{k} r_{TCA} \implies \exists (p,k) \in \beta : \exists (r_B, r_{TCA}) \in R : q_B \xrightarrow[B]{\beta^* p \beta^*} r_B\,.$$

Hereafter, we detail proofs of each of these four points:

(i) If $q_B \xrightarrow[B]{\beta} r_B$, then by definition (7) of the relation β, the corresponding transition is either labelled by an internal port or by a real number representing a delay transition. Note that if β corresponds to an internal port $p \in P_l \cap P_i$, by definition (8) of the relation R, we have $n \in \{N_l^0, l_p\}$ (see Figure 9b), $v_c(q_B) = v_c^*(q_{TCA})$ and $v_x(q_B) = v_x^*(q_{TCA})$.

Case 1: β corresponds to an internal port $p \in P_l \cap P_i$ and $n = l_p$. By Definition 3, there is a transition $l \xrightarrow{p, g_X, g_C, \emptyset, f} l'$ in B (recall, Section 5, that no clocks are reset in TT-BIP models), with

$$g_X(v_x(q_B)) = g_C(v_c(q_B)) = \mathit{True}, \quad v_x(r_B) = f(v_x(q_B)), \quad \text{and} \quad v_c(r_B) = v_c(q_B)\,. \tag{9}$$

By definition of F (Definition 7), there is a corresponding transition τ_{l_p}:

$$l_p \xrightarrow{\mathit{True}, \big((-1,\perp),(-1,\perp),(-1,\perp)\big),(p,\mathit{internal}),f} N^0_{l'}$$

in $F(B)$. By construction (8) of R, we have $q_{TCA} = (l_p, v_x(q_{TCA}), v_c(q_{TCA}), v_{\lambda_{ref}}(q_{TCA}))$, such that

$$v_c(q_B) = v^*_c(q_{TCA}) \quad \text{and} \quad v_x(q_B) = v^*_x(q_{TCA})\,. \tag{10}$$

Therefore, by definition of G_2 (Definition 5), we also have $q_{TCA} \xrightarrow[TCA]{(p,\mathit{internal})} r_{TCA}$, where $r_{TCA} = (N^0_{l'}, v_x(r_{TCA}), v_c(r_{TCA}), v_{\lambda_{ref}}(r_{TCA}))$, with

$$v_c(r_{TCA}) = v_c(q_{TCA}),\ v_{\lambda_{ref}}(r_{TCA}) = v_{\lambda_{ref}}(q_{TCA}) \text{ and } v^*_x(r_{TCA}) = f(v^*_x(q_{TCA}))\,. \tag{11}$$

(For the latter equality, notice that, for internal ports $p \in P_i$, the function f in the transition τ_{l_p} only operates on variables in X, but not on those in $X' \cup Y$.)

Combining (9), (10) and (11), we obtain that $v^*_c(r_{TCA}) = v_c(r_B)$ and $v^*_x(r_{TCA}) = v_x(r_B)$. Thus, we have $q_{TCA} \xrightarrow[TCA]{\beta} r_{TCA}$ and, by (8), $(r_B, r_{TCA}) \in R$.

Case 2: β corresponds to an internal port $p \in P_l \cap P_i$ and $n = N^0_l$. By Definition 3, there is a transition $l \xrightarrow{p, g_X, g_C, \emptyset, f} l'$ in B (recall, Section 5, that no clocks are reset in TT-BIP models), with $g_C = (lb \leq c \leq ub)$ and

$$g_X(v_x(q_B)) = g_C(v_c(q_B)) = \mathit{True}, \quad v_x(r_B) = f(v_x(q_B)), \quad \text{and} \quad v_c(r_B) = v_c(q_B)\,. \tag{12}$$

By definition of F (Definition 7), there are two corresponding successive transition $\tau^0_{l_p}$ and τ_{l_p}:

$$N^0_l \xrightarrow{g_X, \big((lb,c),(ub,c),(-1,\perp)\big),(p,\mathit{internal}),id} l_p \xrightarrow{\mathit{True}, \big((-1,\perp),(-1,\perp),(-1,\perp)\big),(p,\mathit{internal}),f} N^0_{l'}$$

in $F(B)$. By construction (8) of R, we have $q_{TCA} = (N^0_l, v_x(q_{TCA}), v_c(q_{TCA}), v_{\lambda_{ref}}(q_{TCA}))$, such that

$$v_c(q_B) = v^*_c(q_{TCA}) \quad \text{and} \quad v_x(q_B) = v^*_x(q_{TCA})\,. \tag{13}$$

Therefore, by definition of G_2 (Definition 5), we also have

$$q_{TCA} \xrightarrow[TCA]{(p,\mathit{internal})} q'_{TCA} \xrightarrow[TCA]{(p,\mathit{internal})} r_{TCA}\,,$$

where

$$\begin{aligned} q'_{TCA} &= (l_p, v_x(q'_{TCA}), v_c(q'_{TCA}), v_{\lambda_{ref}}(q'_{TCA}))\,, \\ r_{TCA} &= (N^0_{l'}, v_x(r_{TCA}), v_c(r_{TCA}), v_{\lambda_{ref}}(r_{TCA}))\,, \end{aligned}$$

with

$$\begin{aligned} v_c(r_{TCA}) &= v_c(q'_{TCA}) = v_c(q_{TCA})\,, \\ v_{\lambda_{ref}}(r_{TCA}) &= v_{\lambda_{ref}}(q'_{TCA}) \neq v_{\lambda_{ref}}(q_{TCA})\,, \\ v^*_x(r_{TCA}) &= f(v^*_x(q'_{TCA})) = f(v^*_x(q_{TCA}))\,. \end{aligned} \tag{14}$$

Combining (12), (13) and (14), we obtain that $v^*_c(r_{TCA}) = v_c(r_B)$ and $v^*_x(r_{TCA}) = v_x(r_B)$. Thus, we have $q_{TCA} \xrightarrow[TCA]{\beta\beta} r_{TCA}$ and, by (8), $(r_B, r_{TCA}) \in R$.

Case 3: β is a real number δ (i.e. delay step) and $n = N^0_l$, for some $l \in L$. By Definition 3, there is a tpc constraint on location l in B, $tpc(l) = (c \leq \delta)$. Therefore:

$$v_x(r_B) = v_x(q_B), \quad \text{and} \quad v_c(r_B) = v_c(q_B) + \delta\,. \tag{15}$$

By definition of F (Definition 7), there is a corresponding transition $\tau^{tpc}_l =$:

$$N^0_l \xrightarrow{True,\big((-1,\perp),(\delta,c),(-1,\perp)\big),(p,internal),id} N^0_l$$

in $F(B)$. By construction (8) of R, we have $q_{TCA} = (N^0_l, v_x(q_{TCA}), v_c(q_{TCA}), v_{\lambda_{ref}}(q_{TCA}))$, such that

$$v_c(q_B) = v^*_c(q_{TCA}) \quad \text{and} \quad v_x(q_B) = v^*_x(q_{TCA})\,. \tag{16}$$

Therefore, by definition of G_2 (Definition 5), we also have $q_{TCA} \xrightarrow[TCA]{(p,internal)} r_{TCA}$, where $r_{TCA} = (N^0_l, v_x(r_{TCA}), v_c(r_{TCA}), v_{\lambda_{ref}}(r_{TCA}))$, with

$$v^*_c(r_{TCA}) = v^*_c(q_{TCA}) + \delta,\ v_{\lambda_{ref}}(r_{TCA}) = v_{\lambda_{ref}}(q_{TCA}) \text{ and } v^*_x(r_{TCA}) = v^*_x(q_{TCA})\,. \tag{17}$$

Combining (15), (16) and (17), we obtain that $v^*_c(r_{TCA}) = v_c(r_B)$ and $v^*_x(r_{TCA}) = v_x(r_B)$. Thus, we have $q_{TCA} \xrightarrow[TCA]{\beta} r_{TCA}$ and, by (8), $(r_B, r_{TCA}) \in R$.

(ii) If $(q_B, q_{TCA}) \in R$, $q_{TCA} \xrightarrow[TCA]{\beta} r_{TCA}$, then by definition (7) of the relation β, the transition β is neither labelled by $(p, send)$ nor $(p, receive)$. It can be labelled only by $(p, internal)$. Applying this to the definition (8) of the relation R, we deduce that this transition can be enabled only from control locations N^0_l and l_p, since from control location $N^1_{l_p}$ the unique enabled transition is labelled by $(p, send)$ (cf. Definition 7). Thus this β transition corresponds in $F(B)$ to one of these transitions; τ^{tpc}_l, $\tau^0_{l_p}$, τ_{l_p} if $p \in P_i$ and $\tau^1_{l_p}$ if $p \in P_s \cup P_r$.

Case 1: β corresponds to τ_l^{tpc} in $F(B)$, for some $l \in L$. By definition of G_2 (Definition 5), there is a transition τ_l^{tpc}:

$$N_l^0 \xrightarrow{True,\big((-1,\perp),(\delta,c),(-1,\perp)\big),(p,internal),id} N_l^0$$

in $F(B)$, with

$$v_c(r_{TCA}) = v_c(q_{TCA}) + \delta,\; v_{\lambda_{ref}}(r_{TCA}) = v_{\lambda_{ref}}(q_{TCA}) \text{ and } v_x(r_{TCA}) = v_x(q_{TCA})\,. \quad (18)$$

By definition of F (Definition 7), there is a corresponding tpc constraint $tpc(l) = (c \leq \delta)$, in B. By construction (8) of R, we have $q_B = (l, v_x(q_B), v_c(q_B))$, such that

$$v_c(q_B) = v_c^*(q_{TCA}) \quad \text{and} \quad v_x(q_B) = v_x^*(q_{TCA})\,. \quad (19)$$

Therefore, by definition of G_1 (Definition 3), we also have $q_B \xrightarrow[B]{\delta} r_B$,

where $r_B = (l, v_x(r_B), v_c(r_B))$, with

$$v_x(r_B) = v_x(q_B), \quad \text{and} \quad v_c(r_B) = v_c(q_B) + \delta\,. \quad (20)$$

Combining (18), (19) and (20), we obtain that $v_c^*(r_{TCA}) = v_c(r_B)$ and $v_x^*(r_{TCA}) = v_x(r_B)$. Thus, we have $q_B \xrightarrow[B]{\beta} r_B$ and, by (8), $(r_B, r_{TCA}) \in R$.

Case 2: β corresponds to τ_l^0 in $F(B)$, for some $l \in L$. By definition of G_2 (Definition 5), there is a transition τ_l^0:

$$N_l^0 \xrightarrow{g_X,\big((lb,c),(ub,c),(-1,\perp)\big),(p,internal),id} l_p$$

in $F(B)$, with

$$\begin{aligned}
v_c(r_{TCA}) &= v_c(q_{TCA}) + \delta\,,\\
shift^c_{c_{BASE}}(v_{\lambda_{ref}}(q_{TCA}), lb) &\leq conv^c_{c_{BASE}}(v_c(q_{TCA}) + \delta) \leq shift^c_{c_{BASE}}(v_{\lambda_{ref}}(q_{TCA}), ub)\,,\\
v_{\lambda_{ref}}(r_{TCA}) &\neq v_{\lambda_{ref}}(q_{TCA})\,,\\
g_X(v_x(q_{TCA})) &= True\,,\\
v_x(r_{TCA}) &= v_x(q_{TCA})\,.
\end{aligned} \quad (21)$$

By definition of F (Definition 7), this corresponds to a delay step before executing the transition :

$$l \xrightarrow{g_X,g_C,p,\emptyset,f} l'\,,$$

in B where $g_C = (lb \leq c \leq ub)$. By construction (8) of R, we have $q_B = (l, v_x(q_B), v_c(q_B))$, such that

$$v_c(q_B) = v_c^*(q_{TCA}) \quad \text{and} \quad v_x(q_B) = v_x^*(q_{TCA})\,. \quad (22)$$

Therefore, by definition of G_1 (Definition 3), we also have $q_B \xrightarrow[B]{\delta} r_B$, where $r_B = (l, v_x(r_B), v_c(r_B))$, with

$$v_x(r_B) = v_x(q_B), \quad \text{and} \quad v_c(r_B) = v_c(q_B) + \delta\,. \tag{23}$$

Combining (21), (22), (23), we obtain that $v_c^*(r_{TCA}) = v_c(r_B)$ and $v_x^*(r_{TCA}) = v_x(r_B)$. Thus, we have $q_B \xrightarrow[B]{\beta} r_B$ and, by (8), $(r_B, r_{TCA}) \in R$.

Case 3: β corresponds to τ_{l_p} in $F(B)$ for some $l \in L$ and $p \in P_l \cap P_i$. By definition of G_2 (Definition 5), there is a transition τ_{l_p}:

$$l_p \xrightarrow{True,\left((-1,\perp),(-1,\perp),(-1,\perp)\right),(p,internal),f} N^0_{l'}$$

in $F(B)$, with

$$\begin{aligned} v_c(r_{TCA}) &= v_c(q_{TCA}) + \delta\,, \delta \in \mathbb{Z}_+\,, \\ v_{\lambda_{ref}}(r_{TCA}) &= v_{\lambda_{ref}}(q_{TCA})\,, \\ v_x(r_{TCA}) &= f(v_x(q_{TCA}))\,. \end{aligned} \tag{24}$$

By definition of F (Definition 7), there is a corresponding transition

$$l \xrightarrow{g_X, g_C, p, \emptyset, f} l'$$

, in B, where $tpc(l) = (c \leq ub_1)$ and $tpc(l') = (c \leq ub_2)$. By construction (8) of R, we have $q_B = (l, v_x(q_B), v_c(q_B))$, such that

$$v_c(q_B) = v_c^*(q_{TCA}) \quad \text{and} \quad v_x(q_B) = v_x^*(q_{TCA})\,. \tag{25}$$

Therefore, by definition of G_1 (Definition 3), we also have

$$q_B \xrightarrow[B]{\delta_1} q'_B \xrightarrow[B]{p \in P_i} q"_B \xrightarrow[B]{\delta_2} r_B\,,$$

, where

$$\begin{aligned} q'_B &= (l, v_x(q'_B), v_c(q'_B))\,, \\ q"_B &= (l', v_x(q"_B), v_c(q"_B))\,, \\ r_B &= (l', v_x(r_B), v_c(r_B))\,, \end{aligned}$$

with

$$\begin{aligned} g_X(v_x(q'_B)) = g_C(v_c(q'_B)) &= True\,, \\ v_x(r_B) = v_x(q"_B) = f(v_x(q'_B)) &= f(v_x(q_B))\,, \\ v_c(r_B) &= v_c(q"_B) + \delta_2\,, \delta_2 \leq ub_2\,, \\ v_c(q"_B) = v_c(q'_B) &= v_c(q_B) + \delta_1\,, \delta_1 \leq ub_1\,. \end{aligned} \tag{26}$$

By Lemma 1, $v_c(r_{TCA})$ should perfectly respect constraints of preceding and succeeding transitions, i.e. constraints mapped from transitions in B reaching l and coming out from l'. Thus, we consider $\delta \in \mathbb{Z}_+$, such that:

$$\delta = \delta_1 + \delta_2 \,. \tag{27}$$

Combining (27), (24), (25) and (26), we obtain that $v_c^*(r_{TCA}) = v_c(r_B)$ and $v_x^*(r_{TCA}) = v_x(r_B)$. Thus, we have $q_B \xrightarrow[B]{\beta^*} r_B$ (since $\beta(p) = \emptyset$) and, by (8), $(r_B, r_{TCA}) \in R$.

Case 4: β corresponds to $\tau^1_{l_p}$ in $F(B)$ for some $l \in L$ and $p \in P_l \cap (P_s \cup P_r)$. By definition of G_2 (Definition 5), there is a transition $\tau^1_{l_p}$ in $F(B)$. If $p \in P_s$, we have

$$l_p \xrightarrow{True,\big((-1,\perp),(1,c_{fg}),(1,c_{fg})\big),(p,internal),id} N^0_{l'}$$

If $p \in P_r$, the transition $\tau^1_{l_p}$ is

$$l_p \xrightarrow{\neg g^p_{fresh},\big((-1,\perp),(1,c_{fg}),(1,c_{fg})\big),(p,internal),id} l_p$$

In both cases, we have

$$v_c^*(r_{TCA}) = v_c^*(q_{TCA}),\ v_{\lambda_{ref}}(r_{TCA}) = v_{\lambda_{ref}}(q_{TCA}) + 1 \text{ and } v_x(r_{TCA}) = v_x(q_{TCA}) \tag{28}$$

(Notice that the transition $\tau^1_{l_p}$ increments the valuation of the clock c_{fg}. Recall that this clock is excluded by the valuation v_c^*).

By construction (8) of R, we have $q_B = (l, v_x(q_B), v_c(q_B))$, such that

$$v_c(q_B) = v_c^*(q_{TCA}) \quad \text{and} \quad v_x(q_B) = v_x^*(q_{TCA}) \,. \tag{29}$$

Combining (28) and (29), we obtain that $v_c^*(r_{TCA}) = v_c(q_B)$ and $v_x^*(r_{TCA}) = v_x(q_B)$. Thus, we have $q_B \xrightarrow[B]{} q_B$ and, by (8), $(q_B, r_{TCA}) \in R$.

(iii) Let $(q_B, q_{TCA}) \in R$ such that $q_B \xrightarrow[B]{p} r_B$. If $\beta(p) \neq \emptyset \wedge q_B \xrightarrow[B]{p} r_B$, then by definition (7) of the relation β, $p \in P_l \cap (P_r \cup P_s)$. By definition (8) of the relation R, we have $n \in \{N^0_l, l_p\} \cup \{N^1_{l_p}\}_{p \in P_s}$ (see Figure 9b).

Case 1: $p \in P_l \cap P_s$ and $n = N^0_l$. By Definition 3, there is a transition $l \xrightarrow{p,g_X,g_C,\emptyset,f} l'$ in B (recall, Section 5, that no clocks are reset in TT-BIP models), where $g_C = (lb \leq c \leq ub)$, such that

$$g_X(v_x(q_B)) = g_C(v_c(q_B)) = True, \quad v_x(r_B) = f(v_x(q_B)), \quad \text{and} \quad v_c(r_B) = v_c(q_B) \,. \tag{30}$$

By definition of F (Definition 7), there are three corresponding successive transitions $\tau_{l_p}^0$, $\tau_{l_p}^1$ and τ_{l_p}:

$$N_l^0 \xrightarrow{g_X,\big((lb,c),(ub,c),(-1,\bot)\big),(p,internal),f_{flag}^p} l_p \xrightarrow{True,\big((-1,\bot),(1,c_{fc}),(1,c_{fc})\big),(p,internal),id} N_{l_p}^1$$

$$N_{l_p}^1 \xrightarrow{True,\big((-1,\bot),(-1,\bot),(-1,\bot)\big),(p,send),f} N_{l'}^0$$

in $F(B)$. By construction (8) of R, we have $q_{TCA} = (N_l^0, v_x(q_{TCA}), v_c(q_{TCA}), v_{\lambda_{ref}}(q_{TCA}))$, such that

$$v_c(q_B) = v_c^*(q_{TCA}) \quad \text{and} \quad v_x(q_B) = v_x^*(q_{TCA})\,. \tag{31}$$

Therefore, by definition of G_2 (Definition 5), we also have

$$q_{TCA} \xrightarrow[TCA]{(p,internal)} q'_{TCA} \xrightarrow[TCA]{(p,internal)} q"_{TCA} \xrightarrow[TCA]{(p,send)} r_{TCA}\,,$$

where

$$\begin{aligned} q'_{TCA} &= (l_p, v_x(q'_{TCA}), v_c(q'_{TCA}), v_{\lambda_{ref}}(q'_{TCA}))\,, \\ q"_{TCA} &= (N_{l_p}^1, v_x(q"_{TCA}), v_c(q"_{TCA}), v_{\lambda_{ref}}(q"_{TCA}))\,, \\ r_{TCA} &= (N_{l'}^0, v_x(r_{TCA}), v_c(r_{TCA}), v_{\lambda_{ref}}(r_{TCA}))\,, \end{aligned}$$

with

$$\begin{aligned} v_c^*(r_{TCA}) &= v_c^*(q"_{TCA}) = v_c^*(q'_{TCA}) = v_c^*(q_{TCA})\,, \\ v_{\lambda_{ref}}(r_{TCA}) &= v_{\lambda_{ref}}(q"_{TCA}) = v_{\lambda_{ref}}(q'_{TCA}) + 1 \neq v_{\lambda_{ref}}(q'_{TCA})\,, \\ v_x^*(r_{TCA}) &= f(v_x^*(q"_{TCA})) = f(v_x^*(q'_{TCA})) = f(v_x^*(q_{TCA}))\,. \end{aligned} \tag{32}$$

(For the latter equality, notice that, for send ports $p \in P_s$, the function f_{flag}^p in the transition $\tau_{l_p}^0$ operates on variables of X' and the function f in the transition τ_{l_p} only operates on variables of X, but not on those of $X' \cup Y$.)

Combining (30), (31) and (32) we obtain that $v_c^*(r_{TCA}) = v_c(r_B)$ and $v_x^*(r_{TCA}) = v_x(r_B)$. Thus, we have

$$q_{TCA} \xrightarrow[TCA]{\beta} q'_{TCA} \xrightarrow[TCA]{\beta} q"_{TCA} \xrightarrow[TCA]{k} r_{TCA}\,,$$

where $k = (p, send)$ and, by (8), $(r_B, r_{TCA}) \in R$.

Case 2: $p \in P_l \cap P_r$ and $n = N_l^0$. By Definition 3, there is a transition $l \xrightarrow{p,g_X,g_C,\emptyset,f} l'$ in B (recall, Section 5, that no clocks are reset in TT-BIP models), where $g_C = (lb \leq c \leq ub)$, such that

$$g_X(v_x(q_B)) = g_C(v_c(q_B)) = True, \quad v_x(r_B) = f(v_x(q_B)), \quad \text{and} \quad v_c(r_B) = v_c(q_B)\,. \tag{33}$$

By definition of F (Definition 7), there are three corresponding successive transitions $\tau_{l_p}^0$, $\tau_{l_p}^1$ and τ_{l_p}:

$$N_l^0 \xrightarrow{g_X,\big((lb,c),(ub,c),(-1,\perp)\big),(p,internal),id} l_p \xrightarrow{\neg g_{fresh}^p,\big((-1,\perp),(1,c_{fg}),(1,c_{fg})\big),(p,internal),id} l_p$$

$$l_p \xrightarrow{g_{fresh}^p,\big((-1,\perp),(-1,\perp),(-1,\perp)\big),(p,receive),f\circ f_{update}^p} N_{l'}^0$$

in $F(B)$. By construction (8) of R, we have $q_{TCA} = (N_l^0, v_x(q_{TCA}), v_c(q_{TCA}), v_{\lambda_{ref}}(q_{TCA}))$, such that

$$v_c(q_B) = v_c^*(q_{TCA}) \quad \text{and} \quad v_x(q_B) = v_x^*(q_{TCA})\,. \tag{34}$$

Therefore, by definition of G_2 (Definition 5), we also have

$$q_{TCA} \xrightarrow[TCA]{(p,internal)} q'_{TCA} \xrightarrow[TCA]{(p,internal)^*} q"_{TCA} \xrightarrow[TCA]{(p,receive)} r_{TCA}$$

(notice that the second transition can happen more than one time before the guard g_{fresh}^p evaluates to *True*), where

$$\begin{aligned} q'_{TCA} &= (l_p, v_x(q'_{TCA}), v_c(q'_{TCA}), v_{\lambda_{ref}}(q'_{TCA}))\,, \\ q"_{TCA} &= (l_p, v_x(q"_{TCA}), v_c(q"_{TCA}), v_{\lambda_{ref}}(q"_{TCA}))\,, \\ r_{TCA} &= (N_{l'}^0, v_x(r_{TCA}), v_c(r_{TCA}), v_{\lambda_{ref}}(r_{TCA}))\,, \end{aligned}$$

with

$$\begin{aligned} v_c^*(r_{TCA}) &= v_c^*(q"_{TCA}) = v_c^*(q'_{TCA}) = v_c^*(q_{TCA})\,, \\ v_{\lambda_{ref}}(r_{TCA}) &= v_{\lambda_{ref}}(q"_{TCA}) = v_{\lambda_{ref}}(q'_{TCA}) + n \neq v_{\lambda_{ref}}(q'_{TCA}) \text{ and}\,, \\ v_x^*(r_{TCA}) &= f(v_x^*(q"_{TCA})) = f(v_x^*(q'_{TCA})) = f(v_x^*(q_{TCA}))\,, \end{aligned} \tag{35}$$

where n denotes the number of times the transition $\tau_{l_p}^1$ (i.e. reception attempts) has executed. For the last equality of (35), notice that, for receive ports $p \in P_r$, in the transition τ_{l_p}, the function f only operates on variables in X, but not on those in $X' \cup Y$.

Combining (33), (34) and (35) we obtain that $v_c^*(r_{TCA}) = v_c(r_B)$ and $v_x^*(r_{TCA}) = v_x(r_B)$. Thus, we have

$$q_{TCA} \xrightarrow[TCA]{\beta} q'_{TCA} \xrightarrow[TCA]{\beta^*} q"_{TCA} \xrightarrow[TCA]{k} r_{TCA}\,,$$

where $k = (p, receive)$ and, by (8), $(r_B, r_{TCA}) \in R$.

Case 3: $p \in P_l \cap P_s$ **and** $n = l_p$**.** By Definition 3, there is a transition $l \xrightarrow{p, g_X, g_C, \emptyset, f} l'$ in B (recall, Section 5, that no clocks are reset in TT-BIP models), with

$$g_X(v_x(q_B)) = g_C(v_c(q_B)) = \mathit{True}, \quad v_x(r_B) = f(v_x(q_B)), \quad \text{and} \quad v_c(r_B) = v_c(q_B)\,. \tag{36}$$

By definition of F (Definition 7), there are two corresponding successive transitions $\tau^1_{l_p}$ and τ_{l_p}:

$$l_p \xrightarrow{\mathit{True},\big((-1,\perp),(1,c_{fc}),(1,c_{fc})\big),(p,\mathit{internal}),id} N^1_{l_p} \xrightarrow{\mathit{True},\big((-1,\perp),(-1,\perp),(-1,\perp)\big),(p,\mathit{send}),f} N^0_{l'}$$

in $F(B)$. By construction (8) of R, we have $q_{TCA} = (l_p, v_x(q_{TCA}), v_c(q_{TCA}), v_{\lambda_{ref}}(q_{TCA}))$, such that

$$v_c(q_B) = v^*_c(q_{TCA}) \quad \text{and} \quad v_x(q_B) = v^*_x(q_{TCA})\,. \tag{37}$$

Therefore, by definition of G_2 (Definition 5), we also have

$$q_{TCA} \xrightarrow[TCA]{(p,\mathit{internal})} q'_{TCA} \xrightarrow[TCA]{(p,\mathit{send})} r_{TCA}$$

, where

$$\begin{aligned} q'_{TCA} &= (N^1_{l_p}, v_x(q'_{TCA}), v_c(q'_{TCA}), v_{\lambda_{ref}}(q'_{TCA}))\,, \\ r_{TCA} &= (N^0_{l'}, v_x(r_{TCA}), v_c(r_{TCA}), v_{\lambda_{ref}}(r_{TCA}))\,, \end{aligned}$$

with

$$\begin{aligned} v^*_c(r_{TCA}) &= v^*_c(q'_{TCA}) = v^*_c(q_{TCA})\,, \\ v_{\lambda_{ref}}(r_{TCA}) &= v_{\lambda_{ref}}(q'_{TCA}) = v_{\lambda_{ref}}(q_{TCA}) + 1\,, \\ v^*_x(r_{TCA}) &= f(v^*_x(q'_{TCA})) = f(v^*_x(q_{TCA}))\,. \end{aligned} \tag{38}$$

For the latter equality, notice that, for send ports $p \in P_s$, the function f^p_{flag} in the transition $\tau^0_{l_p}$ operates on variables of X' and the function f in the transition τ_{l_p} only operates on variables in X, but not on those in $X' \cup Y$.

Combining (36), (37) and (38) we obtain that $v^*_c(r_{TCA}) = v_c(r_B)$ and $v^*_x(r_{TCA}) = v_x(r_B)$. Thus, we have

$$q_{TCA} \xrightarrow[TCA]{\beta} q'_{TCA} \xrightarrow[TCA]{k} r_{TCA}\,,$$

where $k = (p, \mathit{send})$ and, by (8), $(r_B, r_{TCA}) \in R$.

Case 4: $p \in P_l \cap P_r$ **and** $n = l_p$**.** By Definition 3, there is a transition $l \xrightarrow{p, g_X, g_C, \emptyset, f} l'$ in B (recall, Section 5, that no clocks are reset in TT-BIP models), with

$$g_X(v_x(q_B)) = g_C(v_c(q_B)) = \mathit{True}, \quad v_x(r_B) = f(v_x(q_B)), \quad \text{and} \quad v_c(r_B) = v_c(q_B)\,. \tag{39}$$

By definition of F (Definition 7), there are two corresponding successive transitions $\tau^1_{l_p}$ and τ_{l_p}:

$$l_p \xrightarrow{\neg g^p_{fresh},\left((-1,\perp),(1,c_{fg}),(1,c_{fg})\right),(p,internal),id} l_p \xrightarrow{g^p_{fresh},\left((-1,\perp),(-1,\perp),(-1,\perp)\right),(p,receive),f\circ f^p_{update}} N^0_{l'}$$

in $F(B)$, where $\tau^1_{l_p}$ can be executed n times before τ_{l_p}. By construction (8) of R, we have $q_{TCA} = (l_p, v_x(q_{TCA}), v_c(q_{TCA}), v_{\lambda_{ref}}(q_{TCA}))$, such that

$$v_c(q_B) = v^*_c(q_{TCA}) \quad \text{and} \quad v_x(q_B) = v^*_x(q_{TCA})\,. \tag{40}$$

Therefore, by definition of G_2 (Definition 5), we also have

$$q_{TCA} \xrightarrow[TCA]{(p,internal)^*} q'_{TCA} \xrightarrow[TCA]{(p,receive)} r_{TCA}$$

, where

$$\begin{aligned} q'_{TCA} &= (l_p, v_x(q'_{TCA}), v_c(q'_{TCA}), v_{\lambda_{ref}}(q'_{TCA}))\,,\\ r_{TCA} &= (N^0_{l'}, v_x(r_{TCA}), v_c(r_{TCA}), v_{\lambda_{ref}}(r_{TCA}))\,, \end{aligned}$$

with

$$\begin{aligned} v^*_c(r_{TCA}) &= v^*_c(q'_{TCA}) = v^*_c(q_{TCA})\,,\\ v_{\lambda_{ref}}(r_{TCA}) &= v_{\lambda_{ref}}(q'_{TCA}) = v_{\lambda_{ref}}(q_{TCA}) + n\,,\\ v^*_x(r_{TCA}) &= f(v^*_x(q'_{TCA})) = f(v^*_x(q_{TCA}))\,, \end{aligned} \tag{41}$$

where n denotes the number of times the transition $\tau^1_{l_p}$ (i.e. reception attempts) has executed. For the last equality of (41), notice that, for receive ports $p \in P_r$, in the transition τ_{l_p}, the function f only operates on variables in X, but not on those in $X' \cup Y$.

Combining (39), (40) and (41) we obtain that $v^*_c(r_{TCA}) = v_c(r_B)$ and $v^*_x(r_{TCA}) = v_x(r_B)$. Thus, we have

$$q_{TCA} \xrightarrow[TCA]{\beta^*} q'_{TCA} \xrightarrow[TCA]{k} r_{TCA}\,,$$

where $k = (p, receive)$ and, by (8), $(r_B, r_{TCA}) \in R$.

Case 5: $p \in P_l \cap P_s$ **and** $n = N^1_{l_p}$ By Definition 3, there is a transition $l \xrightarrow{p,g_X,g_C,\emptyset,f} l'$ in B (recall, Section 5, that no clocks are reset in TT-BIP models), with

$$g_X(v_x(q_B)) = g_C(v_c(q_B)) = True, \quad v_x(r_B) = f(v_x(q_B)), \quad \text{and} \quad v_c(r_B) = v_c(q_B)\,. \tag{42}$$

By definition of F (Definition 7), there is a corresponding transition τ_{l_p}:

$$N^1_{l_p} \xrightarrow{True,\left((-1,\perp),(-1,\perp),(-1,\perp)\right),(p,send),f} N^0_{l'}$$

in $F(B)$. By construction (8) of R, we have $q_{TCA} = (N^1_{l_p}, v_x(q_{TCA}), v_c(q_{TCA}), v_{\lambda_{ref}}(q_{TCA}))$, such that

$$v_c(q_B) = v^*_c(q_{TCA}) \quad \text{and} \quad v_x(q_B) = v^*_x(q_{TCA})\,. \tag{43}$$

Therefore, by definition of G_2 (Definition 5), we also have

$$q_{TCA} \xrightarrow[TCA]{(p,send)} r_{TCA}$$

, where

$$r_{TCA} = (N^0_{l'}, v_x(r_{TCA}), v_c(r_{TCA}), v_{\lambda_{ref}}(r_{TCA}))\,,$$

with

$$\begin{aligned} v^*_c(r_{TCA}) &= v^*_c(q_{TCA})\,, \\ v_{\lambda_{ref}}(r_{TCA}) &= v_{\lambda_{ref}}(q_{TCA})\,, \\ v^*_x(r_{TCA}) &= f(v^*_x(q_{TCA}))\,. \end{aligned} \tag{44}$$

(For the latter equality, notice that, for send ports $p \in P_s$, the function f in the transition τ_{l_p} only operates on variables in X, but not on those in $X' \cup Y$.)

Combining (42), (43) and (44) we obtain that $v^*_c(r_{TCA}) = v_c(r_B)$ and $v^*_x(r_{TCA}) = v_x(r_B)$. Thus, we have $q_{TCA} \xrightarrow[TCA]{k} r_{TCA}$, where $k = (p, send)$ and, by (8), $(r_B, r_{TCA}) \in R$.

(iv) If $(q_B, q_{TCA}) \in R$ and $k \in K$ such that $q_{TCA} \xrightarrow[TCA]{k} r_{TCA}$, then by definition (7) of the relation β, $k = (p, send)$ or $k = (p, receive)$. By definition of F (Definition 7), we have $n = l_p$ if $p \in P_l \cap P_r$ and $n = N^1_{l_p}$ if $p \in P_l \cap P_s$.

Case 1: $k = (p, send)$ and $n = N^1_{l_p}$,, for some $l \in L$. By definition of G_2 (Definition 5), there is a transition τ^p_l:

$$N^1_{l_p} \xrightarrow{True,\big((-1,\perp),(-1,\perp),(-1,\perp)\big),(p,send),f} N^0_{l'}$$

in $F(B)$, with

$$\begin{aligned} v_c(r_{TCA}) &= v_c(q_{TCA}) + \delta\,, \delta \in \mathbb{Z}_+\,, \\ v_{\lambda_{ref}}(r_{TCA}) &= v_{\lambda_{ref}}(q_{TCA})\,, \\ v_x(r_{TCA}) &= f(v_x(q_{TCA}))\,. \end{aligned} \tag{45}$$

By definition of F (Definition 7), there is a corresponding transition $l \xrightarrow{g_X,g_C,p,\emptyset,f} l'$, in B where $tpc(l) = (c \leq ub_1)$ and $tpc(l') = (c \leq ub_2)$.. By construction (8) of R, we have $q_B = (l, v_x(q_B), v_c(q_B))$, such that

$$v_c(q_B) = v^*_c(q_{TCA}) \quad \text{and} \quad v_x(q_B) = v^*_x(q_{TCA})\,. \tag{46}$$

Therefore, by definition of G_1 (Definition 3), we also have

$$q_B \xrightarrow[B]{\delta_1} q'_B \xrightarrow[B]{p} q"_B \xrightarrow[B]{\delta_2} r_B \,,$$

, where

$$\begin{aligned} q'_B &= (l, v_x(q'_B), v_c(q'_B))\,, \\ q"_B &= (l', v_x(q"_B), v_c(q"_B))\,, \\ r_B &= (l', v_x(r_B), v_c(r_B))\,, \end{aligned}$$

with

$$\begin{aligned} g_X(v_x(q'_B)) = g_C(v_c(q'_B)) &= \mathit{True}\,, \\ v_x(r_B) = v_x(q"_B) = f(v_x(q'_B)) &= f(v_x(q_B))\,, \\ v_c(r_B) = v_c(q"_B) + \delta_2 &\,, \delta_2 \leq ub_2\,, \\ v_c(q"_B) = v_c(q'_B) = v_c(q_B) + \delta_1 &\,, \delta_1 \leq ub_1\,. \end{aligned} \tag{47}$$

By Lemma 1, $v_c(r_{TCA})$ should perfectly respect constraints of preceding and succeeding transitions, i.e. constraints mapped from transitions in B reaching l and coming out from l'. Thus, we consider $\delta \in \mathbb{Z}_+$, such that:

$$\delta = \delta_1 + \delta_2\,. \tag{48}$$

Combining (48), (45), (46) and (47), we obtain that $v_c^*(r_{TCA}) = v_c(r_B)$ and $v_x^*(r_{TCA}) = v_x(r_B)$. Thus, we have $q_B \xrightarrow[B]{\beta p \beta} r_B$ and, by (8), $(r_B, r_{TCA}) \in R$.

Case 2: $k = (p, receive)$ **and** $n = l_p$,**, for some** $l \in L$**.** By definition of G_2 (Definition 5), there is a transition τ_l^p:

$$l_p \xrightarrow{g^p_{fresh},\big((-1,\perp),(-1,\perp),(-1,\perp)\big),(p,receive),f\circ f^p_{update}} N^0_{l'}$$

in $F(B)$, with

$$\begin{aligned} v_c(r_{TCA}) &= v_c(q_{TCA})\,, \delta \in \mathbb{Z}_+\,, \\ v_{\lambda_{ref}}(r_{TCA}) &= v_{\lambda_{ref}}(q_{TCA})\,, \\ v_x^*(r_{TCA}) &= f(v_x^*(q_{TCA}))\,. \end{aligned} \tag{49}$$

Notice that even if the actual reception was performed in the β transition preceding this k transition, the update of local variables according the received message is only performed via the execution of the k transition (via f^p_{update}). The function f^p_{update} applies to variables of $X \cup Y$. Thus the composition $f \circ f^p_{update}$ preserves the valuation v_x^*.

By definition of F (Definition 7), there is a corresponding transition $l \xrightarrow{g_X, g_C, p, \emptyset, f} l'$, in B where $tpc(l) = (c \leq ub_1)$ and $tpc(l') = (c \leq ub_2)$.. By construction (8) of R, we have $q_B = (l, v_x(q_B), v_c(q_B))$, such that

$$v_c(q_B) = v_c^*(q_{TCA}) \quad \text{and} \quad v_x(q_B) = v_x^*(q_{TCA})\,. \tag{50}$$

Therefore, by definition of G_1 (Definition 3), we also have

$$q_B \xrightarrow[B]{\delta_1} q'_B \xrightarrow[B]{p} q"_B \xrightarrow[B]{\delta_2} r_B \,,$$

where

$$\begin{aligned} q'_B &= (l, v_x(q'_B), v_c(q'_B))\,, \\ q"_B &= (l', v_x(q"_B), v_c(q"_B))\,, \\ r_B &= (l', v_x(r_B), v_c(r_B))\,, \end{aligned}$$

with

$$\begin{aligned} g_X(v_x(q'_B)) = g_C(v_c(q'_B)) &= \mathit{True}\,, \\ v_x(r_B) = v_x(q"_B) = f(v_x(q'_B)) &= f(v_x(q_B))\,, \\ v_c(r_B) &= v_c(q"_B) + \delta_2\,, \delta_2 \le ub_2\,, \\ v_c(q"_B) = v_c(q'_B) &= v_c(q_B) + \delta_1\,, \delta_1 \le ub_1\,. \end{aligned} \tag{51}$$

By Lemma 1, $v_c(r_{TCA})$ should perfectly respect constraints of preceding and succeeding transitions, i.e. constraints mapped from transitions in B reaching l and coming out from l'. Thus, we consider $\delta \in \mathbb{Z}_+$, such that:

$$\delta = \delta_1 + \delta_2\,. \tag{52}$$

Combining (52), (49), (50) and (51), we obtain that $v_x^*(c_{TCA}) = v_c(r_B)$ and $v_x^*(r_{TCA}) = v_x(r_B)$. Thus, we have $q_B \xrightarrow[B]{\beta p \beta} r_B$ and, by (8), $(r_B, r_{TCA}) \in R$.

□

7 Compatibility with composition

In Section 6, we prove that the transformation of individual TT-BIP components into TCA automata is semantics-preserving. In this section, we explain why the composition of all obtained TCA automata is equivalent to the initial TT-BIP model.

Both glues of TCA automata and TT-BIP components provide the same unidirectional transfer of data. The unique difference is that in TT-BIP, interactions provide synchronisation on top of data transfer while in TCA the communication is asynchronous. Constraints, necessary to make synchronizations possible, are reflected in the time constraints of individual components of the TT-BIP model. The transformation from TT-BIP to TCA —described in Section 5.2—ensures that these synchronization constraints are respected in the obtained automaton. Asynchronous (sending and receiving) actions between interacting TCA automata are ensured to happen at instants over a finer-grained clock as described in third paragraph of Section 5.1. With respect to the clock over which the initial synchronization date is defined, these actions are happening at the same instant.

Hence, the correctness of the TCA composition (after step 2 transformation) follows from the correctness of the the transformation of individual components of TT-BIP model.

8 Case study

In order to illustrate the transformation from TT-BIP to TCA presented in the paper, we use the medium voltage protection relay application of [15] as a case study.

A protection relay is a device designed to detect and isolate faults in an electrical network. A sensor measures the current that flows on the network and transmits this information to the relay. The relay receives this information, applies signal processing algorithms and protection algorithms and takes control decisions.

The medium voltage protection relay [15] is first modelled using the RT-BIP framework. Then we apply the transformation of [11] in order to obtain its corresponding TT-BIP model. And finally we apply the transformation described in Section 5.2 in order to generate its corresponding TCA automata. In order to evaluate the generated application, we compare its traces and some other features (development time, size etc.) with those of the manually written version (i.e. PsyC code) of the application [15]. This comparison is relevant since PsyC code corresponds to the same abstraction level as generated TCA automata.

8.1 RT-BIP Modelisation

A model of the application written in RT-BIP is represented by Figure 11. We can divide the model in three stages: acquisition, measurement and protection stages.

The acquisition stage. It collects data and makes them available to the other components of the system. Data are periodically collected every 555 μs. The component who performs this collect and makes data available for the other component is called `Acquisition`.

The measurement stage. It computes different values that will be used in the protection stage. In this case, three different values are computed. The average value is computed by the `Average` component and consists in the computation of the average of the last three values acquired by the `Acquisition` component. This component produces value every time the `Acquisition` component acquires three new data, (i.e. every 1.665 ms). The crest value is computed by the `Crest` component and consists in the computation of the crest value of every value acquired by the `Acquisition` component. This value is computed for every data acquired by the `Acquisition` (i.e every 555 μs). The computation of the magnitude of the fundamental and some harmonics is made by the `TRS` component. This latter uses the last 12 values computed by the `average` component and the last value of the `Crest` component. New values are computed every 12 new data (i.e. every 6.660 ms).

The protection stage. It detects failure by using different algorithms. In this model, we consider two protection algorithms: a instantaneous over-current protection called Protection 50 and an inverse time over-current protection called Protection 51. The protection stage is made of one component for each protection. They check if the safety function of the protection relay must be activated whenever they receive data from the `TRS component` (i.e., every 6.660 ms).

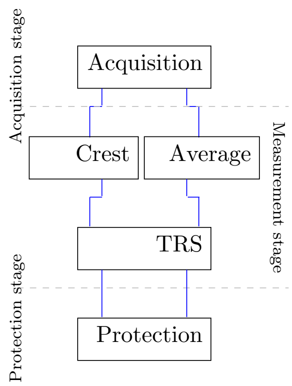


Figure 11: RT-BIP example

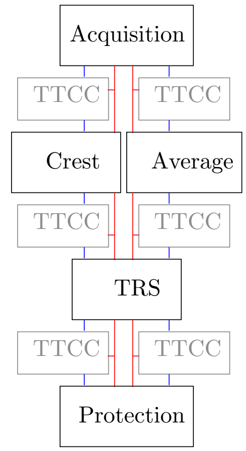


Figure 12: TT-BIP example

8.2 RT-BIP to TT-BIP transformation

We have applied the automatic transformation of [11] in order to obtain the TT-BIP model from the RT-BIP model of the case study. We chose to gather the two protection components in the same task. The rest of components are considered as independent tasks. The resulting TT-BIP model is shown in Figure 12. We have also observed identical values of the output flows generated by simulations –in BIP environment– of both RT-BIP and TT-BIP models.

8.3 TT-BIP to PharOS implementation

We have applied the implemented transformation described in Section 5 to the TT-BIP model of the case-study described above. For each component in the TT-BIP model, we generate an agent in PharOS. Communication between different component is performed through `advance` nodes.

For evaluation purposes, some minor optimisations have been manually made on the generated code. These optimizations solely consist in removing the "empty" generated transitions: those with a constant guard *True*, label $\big((\text{-}1, \emptyset), (\text{-}1, \emptyset), (\text{-}1, \emptyset)\big)$ and without an update function. This optimization does not impact the performance of the generated code, nor the correctness of the proofs. Although it could have been easily automated and formalized, that have not been performed in order to simplify the presentation and the tool prototypes.

Preservation of the functional behaviour of each generated agent (compared to its corresponding component in the TT-BIP model), has been tested as well. We have also observed identical values of the output flows generated by simulations in both environments.

8.4 Evaluation

In this subsection, we compare the automatically generated code with a manually written one [15] for the same case study (cf. Table 1).Notice that with the implemented code generation tool we gain in terms of development time, even if in the present state, we need to adapt the generated code manually since some features are still not included in the implemented tool

	Manually written code	Generated code
Development time	2-3 months	1 week (RT-BIP model writing and validation) + 2 days (code adaptation)
Text section size	41.7 kB	71.2 kB
Application text section size (w/o kernel)	13.9 kB	37.1 kB
Data section size	22.1 kB	31.1 kB
Number of *Temporal variables*	7	18

Table 1: Comparison between the generated and the manually-written source codes of the case study

(e.g. optimisations). In the generated code we introduce almost two and a half times more *temporal variables* compared to the initial model, this is due to the communication atomicity breaking brought by the transformation from RT-BIP model into a TT-BIP model. These added *temporal variables* lead to a larger memory footprint. When comparing *text* and *data* segments sizes with the manually written version, we find out that segments of the automatically generated code have almost two times bigger size. This ratio is rather reasonable and very encouraging as we are not (yet) interested in optimizing the output model in terms of the number of agents and communications. A comparison of the temporal evolution of computed variables in both versions is also of interest. In Figure 13, we display the evolution of the variables *arga* and *crest* in both versions. Values of variable *arga* are transmitted by the sensor to the `acquisition` component, standing for the measures of the input current. *crest* values are computed by the `crest` component. In Figure 13, solid lines are reserved to the automatically generated application, and dotted lines are reserved for the manually written one.

Visual inspection of different values of both variables in both versions, reveals that the output of the automatically generated model is strictly similar to that of the manual model.

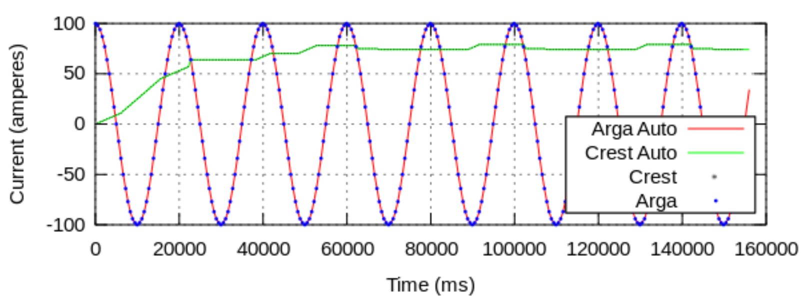


Figure 13: Execution trace

The evaluation of the generated code in terms of CPU overhead compared to the manually written code, is subject of ongoing work.

9 Discussion & conclusion

In this paper, we have detailed our approach to generate correct-by-construction TT implementations from high-level RT-BIP models. This transformation is divided into two steps. First, RT-BIP model is transformed in order to express intertask communication according to the TT communication paradigm. Then, the obtained model is transformed into TCA automata (the defined computation model of PharOS applications). The first step was described in our previous work [11]. In this paper, we have presented the transformation of the second step. First, we have discussed various challenges of this transformation, encoded both initial and final models into a common semantic model (LTS) and presented in details the formal transformation rules. We further have proved the correctness of this transformation using the notion of observational equivalence. Details about the implemented tool for automatic transformation have not been presented in this paper due to the lack of space. Our experiments on an industrial case study show highly encouraging results.

We are planning to pursue with an exhaustive evaluation of the generated code of the case study (CPU overhead, etc.). Furthermore, we have identified several open challenges that we believe should be addressed in future work:

- **Identification of OS service patterns potentially existing in the initial model**: any OS has a number of services (communication, synchronization, etc.). We strongly think that in some initial models, and in components intended for handling communication, we can identify exactly the same behavioural pattern of one or more OS services. Transformation should take this redundancy into account, and only transform into TCA automata the part of the component which can not be mapped to an OS service. The identified pattern is thus mapped to a system call.

- **What about a generic transformation process?** We strongly believe that the transformation process defined above can be generalised to any RTOS-based implementation approach with TT execution model. In fact, we just need to present the semantics of the computation model of the target platform as an LTS system.

References

[1] Tesnim Abdellatif. *Rigourous Implementation of real-time systems.* PhD thesis, UJF, 2012.

[2] Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.

[3] C Aussagues, D Chabrol, V David, D Roux, N Willey, A Tournadre, and M Graniou. PharOS, a multicore OS ready for safety-related automotive systems: results and future prospects. *Proc. of The Embedded Real-Time Software and Systems (ERTS2)*, 2010.

[4] Simon Bliudze, Xavier Fornari, and Mathieu Jan. From model-based to real-time execution of safety-critical applications: Coupling SCADE with OASIS. In *Embedded Real Time Software and Systems*, ERTS2, page 10 pages, February 2012.

[5] Borzoo Bonakdarpour, Marius Bozga, Mohamad Jaber, Jean Quilbeuf, and Joseph Sifakis. From high-level component-based models to distributed implementations. In *Proceedings of the tenth ACM international conference on Embedded software*, pages 209–218. ACM, 2010.

[6] Etienne Borde, Smail Rahmoun, Fabien Cadoret, Laurent Pautet, Frank Singhoff, and Pierre Dissaux. Architecture models refinement for fine grain timing analysis of embedded systems. In *Rapid System Prototyping (RSP), 2014 25th IEEE International Symposium on*, pages 44–50. IEEE, 2014.

[7] Jean-Louis Boulanger, François-Xavier Fornari, Jean-Louis Camus, and Bernard Dion. *SCADE: Language and Applications*. Wiley-IEEE Press, 1st edition, 2015.

[8] Paraskevas Bourgos. *Rigorous Design Flow for Programming Manycore Platforms*. PhD thesis, Grenoble, 2013.

[9] Fabien Cadoret, Etienne Borde, Sebastien Gardoll, and Laurent Pautet. Design patterns for rule-based refinement of safety critical embedded systems models. In *Engineering of Complex Computer Systems (ICECCS), 2012 17th International Conference on*, pages 67–76. IEEE, 2012.

[10] Paul Caspi, Adrian Curic, Aude Maignan, Christos Sofronis, Stavros Tripakis, and Peter Niebert. From Simulink to SCADE/Lustre to TTA: A layered approach for distributed embedded applications. *ACM Sigplan Notices*, 38(7):153–162, 2003.

[11] Hela Guesmi, Belgacem Ben Hedia, Simon Bliudze, Saddek Bensalem, and Jacques Combaz. Towards time-triggered component-based system models. In *ICSEA15*, pages 157–169, Barcelone, Spain, November 2015. ThinkMind.

[12] Hela Guesmi, Belgacem Ben Hedia, Simon Bliudze, Mathieu Jan, and Saddek Bensalem. Towards Correct Transformation: From High-Level Models to Time-Triggered Implementations. In *RTAS*, Work-in-Progress and Demo Proceeding, page 13, Vienna, Austria, April 2016.

[13] Hela Guesmi, Belgacem Ben Hedia, Simon Bliudze, Mathieu Jan, Saddek Bensalem, and Briag Lenabec. Correct transformation of high-level models into time-triggered implementations. Technical Report EPFL-REPORT-218459, EPFL, May 2016.

[14] Nicholas Halbwachs, Paul Caspi, Pascal Raymond, and Daniel Pilaud. The synchronous data flow programming language LUSTRE. *Proceedings of the IEEE*, 79(9):1305–1320, 1991.

[15] Mathieu Jan, Vincent David, Jimmy Lalande, and Maurice Pitel. Usage of the safety-oriented real-time OASIS approach to build deterministic protection relays. In 5^{th} *Intl. Symp. on Industrial Embedded Systems (SIES 2010)*, pages 128–135, Univ. of Trento, 2010.

[16] Robert Kaiser and Stephan Wagner. Evolution of the PikeOS microkernel. In *Proceedings of the 1st International Workshop on Microkernels for Embedded Systems*, pages 50–57, 2007.

[17] Hermann Kopetz. The time-triggered approach to real-time system design. *Predictably Dependable Computing Systems. Springer*, 1995.

[18] Hermann Kopetz and Günther Bauer. The time-triggered architecture. *Proceedings of the IEEE*, 91(1):112–126, 2003.

[19] Matthieu Lemerre, Vincent David, Christophe Aussaguès, and Guy Vidal-Naquet. An introduction to time-constrained automata. *arXiv preprint arXiv:1010.5571*, 2010.

[20] Robin Milner. *Communication and Concurrency*. Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK, 1995.

[21] Kathy Dang Nguyen, PS Thiagarajan, and Weng-Fai Wong. A UML-based design framework for time-triggered applications. In *Real-Time Systems Symposium (RTSS 2007)*, pages 39–48. IEEE, 2007.

[22] Traian Pop, Paul Pop, Petru Eles, Zebo Peng, and Alexandru Andrei. Timing analysis of the FlexRay communication protocol. *Real-time systems*, 39(1-3):205–235, 2008.

[23] Ahlem Triki, Borzoo Bonakdarpour, Jacques Combaz, and Saddek Bensalem. Automated conflict-free concurrent implementation of timed component-based models. In *NASA Formal Methods*, pages 359–374. Springer, 2015.